

PROJECT REPORT

INVENTORY MANAGEMENT SYSTEM FOR RETAILERS

DATE : 18/11/2022

TEAM ID : PNT2022TMID34246

BATCH NO : B12-6A2E

**TEAM MEMBERS : HEERTTIKA S
(960519104029)
BEJANSHINI B
(960519104015)
DIVYA A
(960519104021)
ESAIVENI E
(960519104024)**

I Introduction

1.1.1.1 Project Overview

1.1.1.2 Purpose

II Literature Survey

2.1 Existing Problem

2.2 Problem Statement Definition

III Ideation & Proposed Solution

3.1 Empathy Map

3.2 Ideation & Brainstorming

3.3 Proposed Solution

3.4 Problem Solution Fit

IV Requirement Analysis

4.1 Functional Requirements

4.2 Non-Functional Requirements

V Project Design

5.1 Data Flow Diagram

5.2 Solution & Technical Architecture

5.3 User Stories

VI Project Planning and Scheduling

6.1 Sprint planning and Estimation

6.2 Milestone And Activities

VII Coding and Solutioning

7.1 Feature I

7.2 Feature II

7.3 Code

7.4. Outputs

VIII Result

IX Advantages and Disadvantages

X Conclusion

XI Future Scope

XII References

1. INTRODUCTION

1.1 Project Overview

The objective of this system is to manage the items in an inventory such as tracking orders, placing orders to other suppliers and checking the items in the inventory. The system allows the admin to maintain the items in the inventory.

Whenever the item levels go low, the system places an order to the supplier. The supplier gets the notification of these orders as soon as they are placed and can send the items to the inventory. There are two login pages each for the admin and supplier.

The software has been developed using the most powerful and secured backend Python and IBM Cloud for the databases and most widely accepted frontend JavaScript with HTML and CSS coding

1.2 Purpose

The primary purpose of inventory management is to ensure there is enough goods or materials to meet demand without creating overstock, or excess inventory

Retail management refers to the process of helping customers find products in your store. It includes everything from increasing your customer pool to how products are presented, and how you fulfill a customer's needs. A good store manager helps customers leave the store with a smile.

2. LITERATURE SURVEY

2.1 Existing problem

- The problem faced by the company is they do not have any systematic system to record and keep their inventory data. It is difficult for the admin to record the inventory data quickly and safely because they only keep it in the logbook and not properly organized.
- Good planning and sales forecast before setting optimal inventory levels, appropriate inventory management requires close coordination between the areas of sales, purchasing and finance.

2.2 Problem Statement Definition

Retail inventory management works by creating systems to log products, receive them into inventory, track changes when sales occur, manage the flow of goods from purchasing to final sale and check stock counts.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes. It is a useful tool to help teams better understand their users. Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.

Inventory Management System For Retailers




3.2 Ideation & Brainstorming

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions.

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

Step-1: Team Gathering, Collaboration and Select the Problem Statement

Template



Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

🕒 10 minutes to prepare
👥 1 hour to collaborate
👤 2-8 people recommended

[Share template feedback](#)

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

🕒 10 minutes

A Team gathering

Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B Set the goal

Think about the problem you'll be focusing on solving in the brainstorming session.

C Learn how to use the facilitation tools

Use the Facilitation Superpowers to run a happy and productive session.

[Open article](#) →

1 Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

🕒 5 minutes

Problem

The problem faced by the company is they do not have any systematic system to record and keep their inventory data. It is difficult for the admin to record the inventory data quickly and safely because they only keep it in the logbook and not properly organized.

Key rules of brainstorming

To run an smooth and productive session

- 👤 Stay in topic.
- 💡 Encourage wild ideas.
- ⏸️ Defer judgment.
- 👂 Listen to others.
- 🗣️ Go for volume.
- 👁️ If possible, be visual.

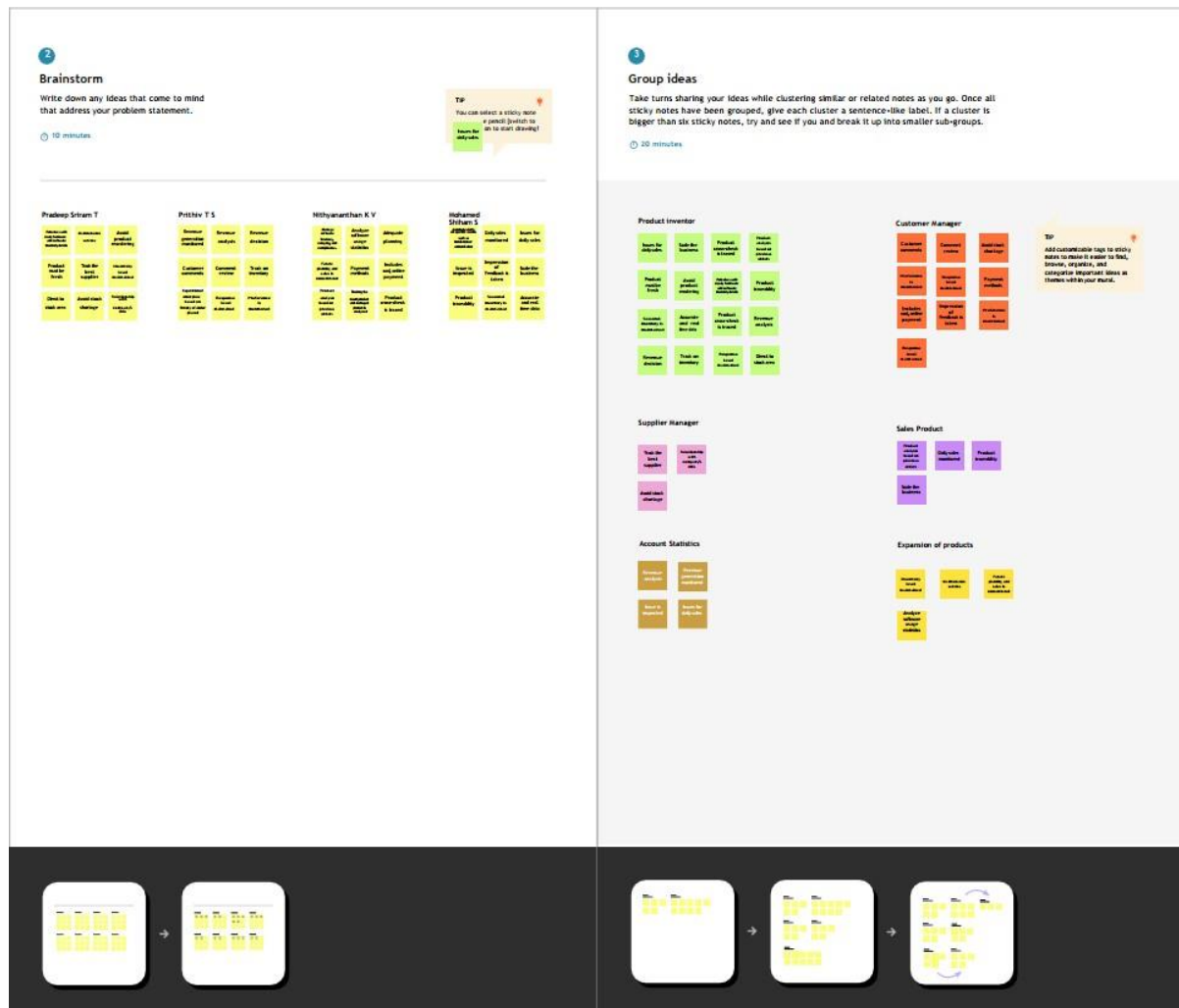


Need some inspiration?

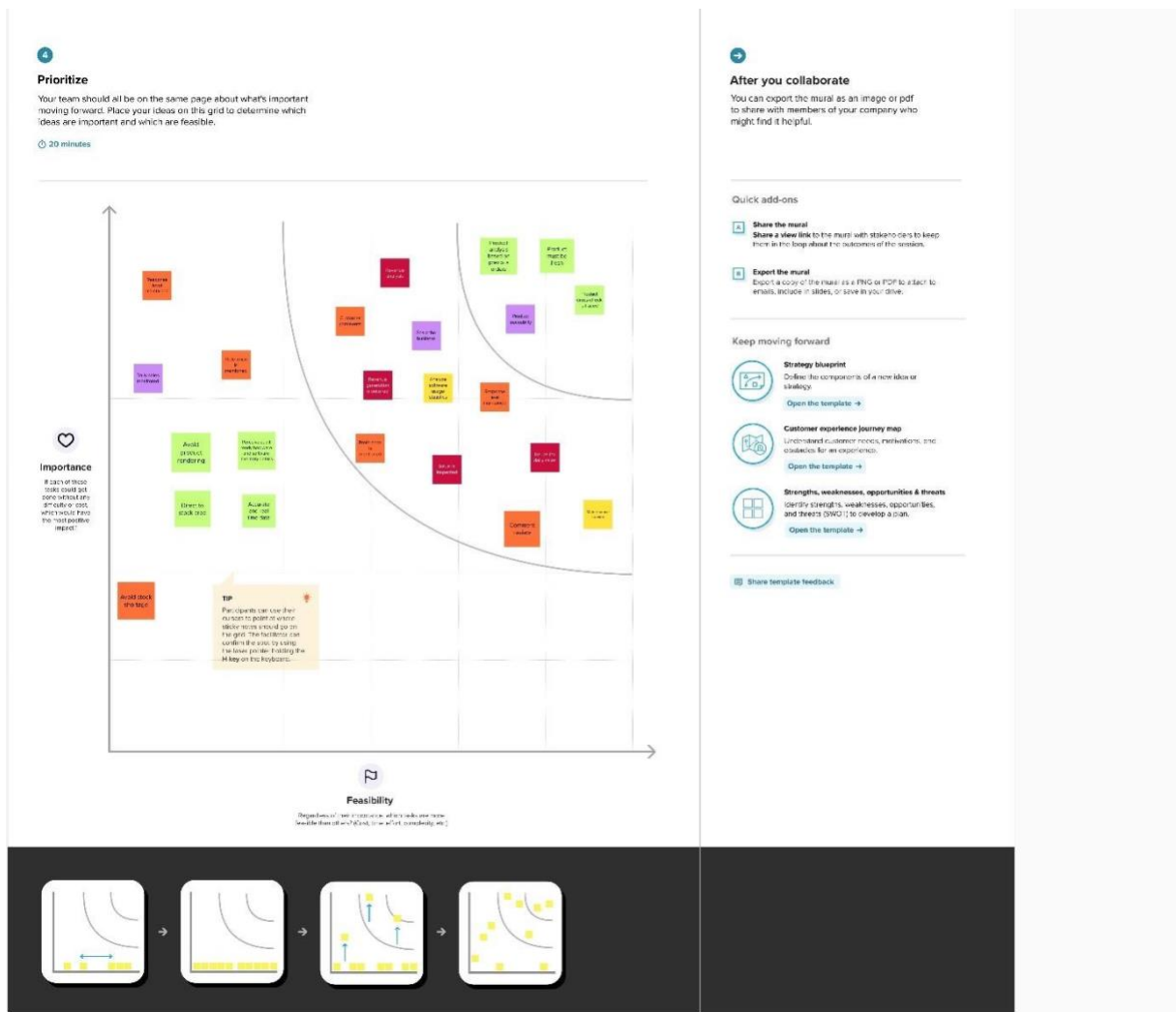
See a finished version of this template to kickstart your work.

[Open example](#) →

Step-2: Brainstorm, Idea Listing and Grouping



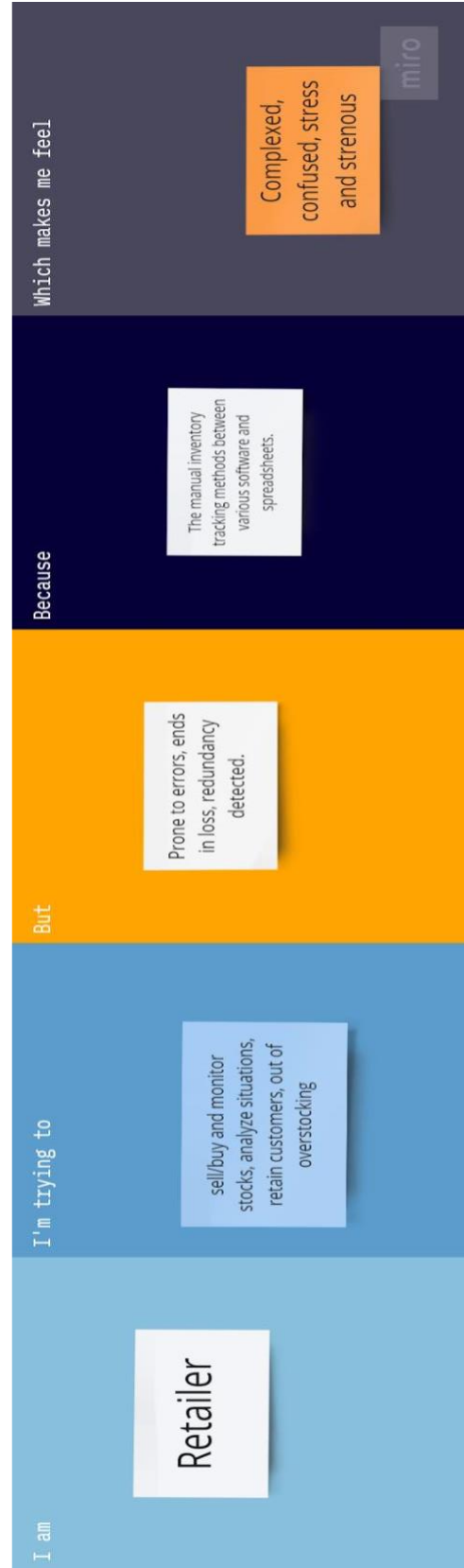
Step-3: Idea Prioritization



3.4 Problem Solution fit



3.5 Customer Problem Statement



4. REQUIREMENT ANALYSIS

4.1 Functional requirement

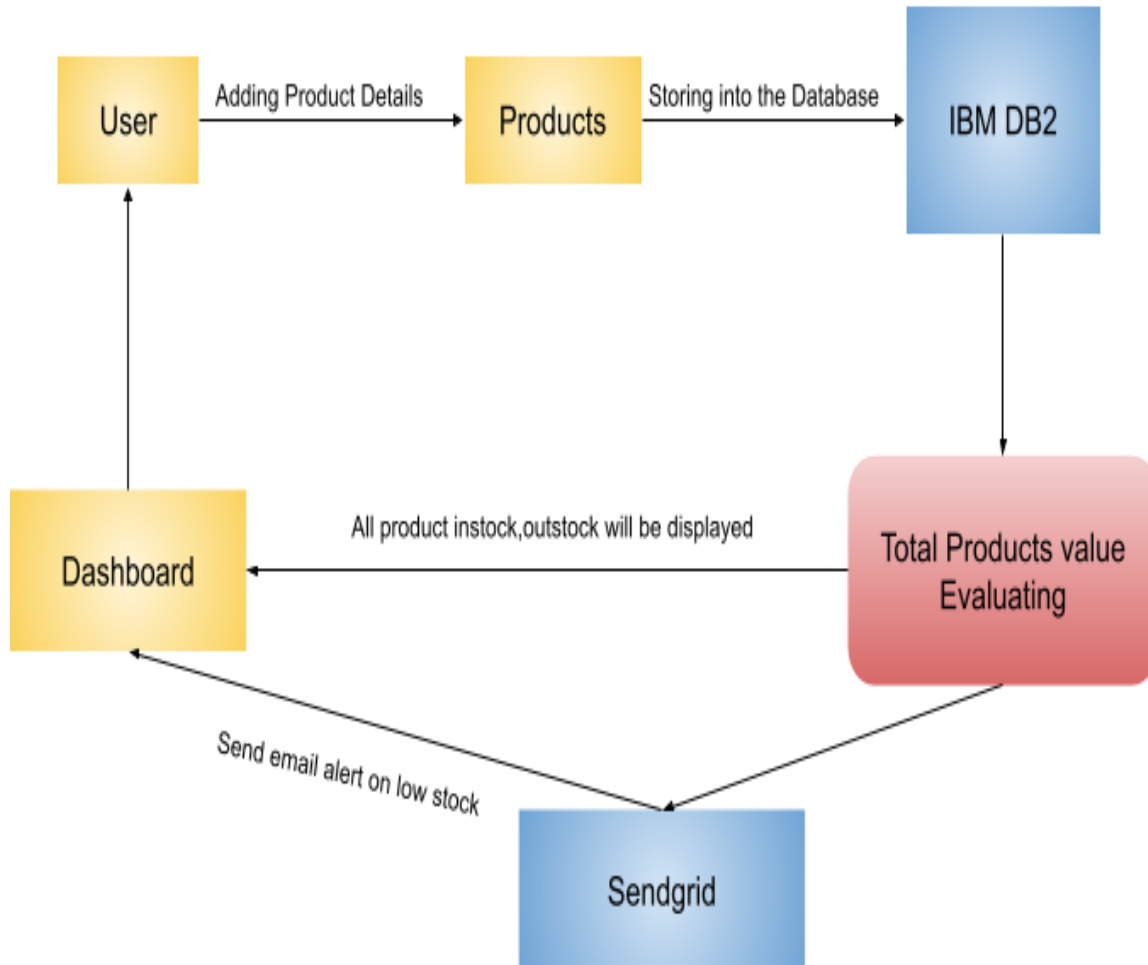
- The System aims at providing an efficient interface to the user for managing of inventory, it shall also provide the user varied options for managing the inventory through various functions at hand. The ingredient levels are continuously monitored based on their usage and are checked for the threshold levels in the inventory and accordingly the user is alerted about low levels of certain ingredients. The design is such that the user does not have to manually update the inventory every time, the System does it for the user.
- The System calculates and predicts the amount of usage for specific set days that are pre-set by the user(admin) , it also alerts the user of an impending action to order ingredients before the specific day set by the user. Therefore the user never has to worry about manually calculating the estimated usage of the ingredients as the System does it for the user.
- The simple interface of the System has functions like adding a recipe, removing or updating the recipe. It also extends to functions such as adding a vendor for an ingredient,, removing the vendor, checking threshold levels, processing orders, altering processed orders etc.

4.2 Non-Functional requirements

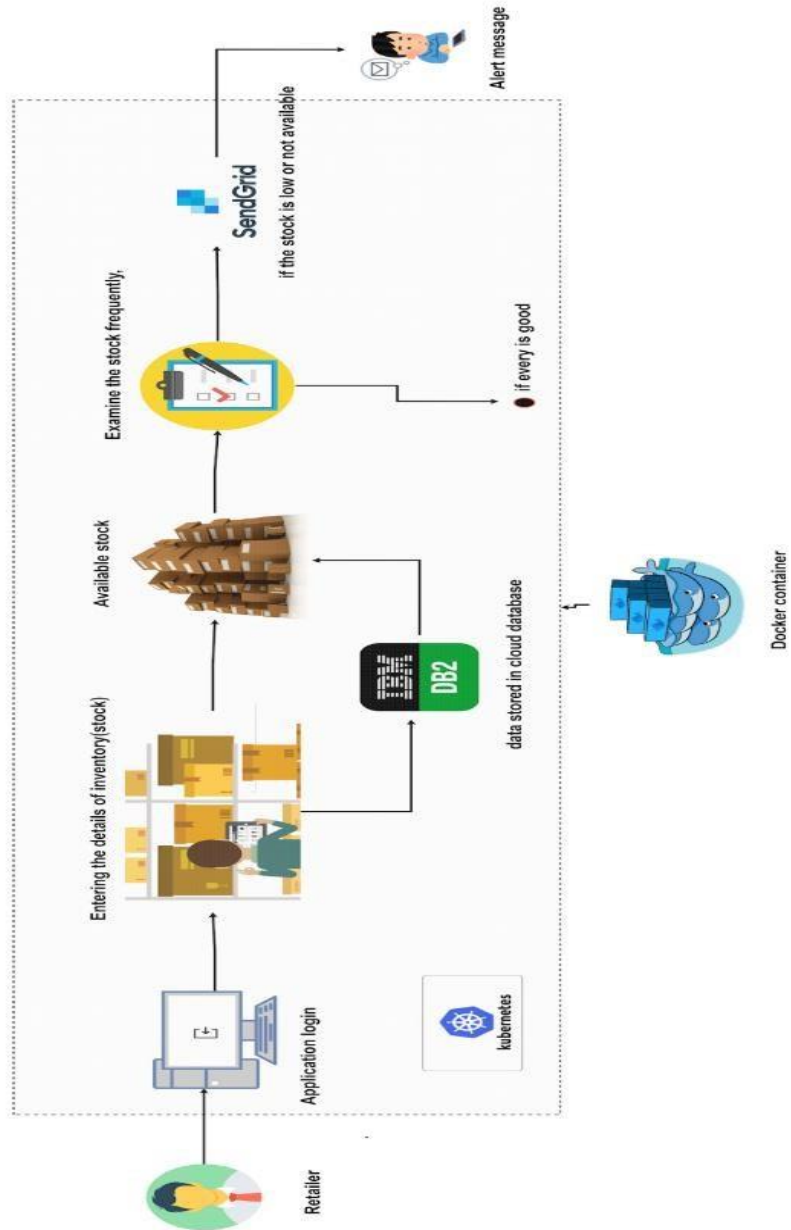
- The system must not lag, because the workers using it don't have down-time to wait for it to complete an action.
- The system must complete updating the databases, adding of recipe, ingredient, vendor and occasions successfully every time the user requests such a process.
- All the functions of the system must be available to the user every time the system is turned on.
- The calculations performed by the system must comply according to the norms set by the user and should not vary unless explicitly changed by the user
- The System must give accurate inventory status to the user continuously. Any inaccuracies are taken care by the regular confirming of the actual levels with the levels displayed in the system.
- The System must successfully add any recipe, ingredients, vendors or special occasions given by the user and provide estimations and inventory status in relevance with the newly updated entities.

5. PROJECT DESIGN

5.1 Data Flow Diagrams



5.2 Solution & Technical Architecture



5.3 User Stories

User Type	Functional Requirement(Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Retailer	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	Medium	Sprint-1
	Login	USN-3	As a user, I can log into the application by entering email & password	I can access my account / dashboard	High	Sprint-1
	Dashboard	USN-4	As a user, I can view the stock list and suppliers list	Once I log in to the system, I can able to view the stocks	Medium	Sprint-1
	Items	USN-5	As a user, I can add the items.	I can create a new type of item	High	Sprint-2
		USN-6	As a user, I can see the items	I can be able to see the items that can be added to the inventory	Low	Sprint-2

	Inventory	USN-7	As a user, I can add the items to inventory.	I can add items to the inventory with quantity	High	Sprint-2
		USN-8	As a user, I can see the items in the inventory.	I can see the inventory items with quantity	Low	Sprint-2
	Indication	USN-9	As a user, I can be able to receive indication	I receive a notification when the stock running low	High	Sprint-3
	Location	USN-10	As a user, I can be able to see items from a particular store location	I can be able to make purchase from a particular location	Medium	Sprint-3
		USN-11	As a user, I can add a new location of my store	I can be able to add new store locations	Medium	Sprint - 3
Customer	Purchase	USN -12	As a customer, I can be able to purchase good from the particular location of the store	I can able to purchase from the store	High	Sprint - 4
Retailer & Customer	Deployment	USN-13	As a user, I can access the software in the web	I can access the software in web	High	Sprint - 4

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	
Sprint-1	Registration	USN-1	As a user, I can register for the application by using my email & password and confirming my login credentials.	3	
Sprint-1		USN-2	As a user, I can login through my E-mail.	3	
Sprint-1	Confirmation	USN-3	As a user, I can receive my confirmation email once I have registered for the application.	2	
Sprint-1	Login	USN-4	As a user, I can log in to the authorized account by entering the registered email and password.	3	
Sprint-2	Dashboard	USN-5	As a user, I can view the products that are available currently.	4	
Sprint-2	Stocks update	USN-6	As a user, I can add products which are not available in the inventory and restock the products.	3	

Sprint-3	Sales prediction	USN-7	As a user, I can get access to sales prediction tool which can help me to predict better restock management of product.	6
Sprint-4	Request for customer care	USN-8	As a user, I am able to request customer care to get in touch with the administrators and enquire the doubts and problems.	4
Sprint-4	Giving feedback	USN-9	As a user, I am able to send feedback forms reporting any ideas for improving or resolving any issues I am facing to get it resolved.	3

6.2 Milestone And Activities

TITLE	DESCRIPTION
Literature Survey & Information Gathering	Literature survey on selected project and gathering information by referring the project's related technical papers, research publications, etc.
Prepare Empathy Map	Prepare empathy map canvas to capture the user's pains & gains and prepare the list of problem statements.

Ideation	To list by the organizing brainstorm sessions and prioritize the top three ideas based on the feasibility and importance.
Proposed Solution	To prepare the proposed solution documents, which includes the novelty, feasibility of ideas, business model, social impact, scalability of the solution, etc.
Problem Solution Fit	Preparing the problem solution fit document.
Solution Architecture	To prepare the solution architecture document
Customer Journey	Prepare the customers journey map help the customers understand the user interaction and experiences with the application from the beginning to the end.
Functional Requirement	Prepare the functional requirement document.
Data Flow Diagrams	Draw the data flow diagrams and submit for the review.
Technology Architecture	Prepare technical architecture diagram.

Prepare Milestone & Activity List	Prepare the milestones and activity of the project.
Project Development – Delivery of Sprint-1, 2, 3 & 4	Develop and submit the developed code by testing it and having no errors.

7. CODING&SOLUTIONING

(Explain the features added in the project along with code)

7.1 Feature 1

Complete insights into key products and service drivers. With the help of tables and symbols, marketers can effectively track and analyse factors that have an effect on important bottom lines like profitability. Store Managers can also effectively optimise product mix across channels, lines and brands with the product scorecards available. Some of the different KPIs that managers can avail of from product performance metrics are product sales by region, change in sales and margin per product, ROI per product, top competitor by product category and much more..

7.2 Feature 2

The entire organisation can access the same store data simultaneously and thus everyone has an understanding of what the customer wants. Managers can better monitor progress, respond immediately to customer needs, adjust parameters for continuous improvement, and exercise greater control over the organisation.

One can record and analyze inventory results and merchandise processes daily to know whether business decisions are based on timely, accurate information.

7.3 Code

Login.html

```
<!{% extends 'base.html' %}
```

```
{% block head %}
```

```
    <title>Login</title>
```

```
{% endblock %}
```

```
{% block body %}
```

```
{{ msg }}
```

```
    <style>
```

```
.divi
```

```
der:
```

```
afte
```

```
r,
```

```
.divi
```

```
der:
```

```
bef
```

```
ore
```

```
{
```

```
con
```

```
tent
```

```
: "";
```

```
flex
```

```
:
```

```
1;
```

```
height:
```

```
1px;
```

```
background
d: #eee;

}

.h-custom { height:
calc(100% - 73px);
}

@media (max-width: 450px) {
```

```
.h-
cu
sto
m
{
hei
gh
t:
10
0%
;
}
}
```

```
.hlink{
textdeco
ration:
none;
}
</style>
```

```

<section class="vh-100">

  <div class="container-fluid h-custom">

    <div class="row d-flex justify-content-center align-items-center h-100">

      <div class="col-md-9 col-lg-6 col-xl-5">

      </div>

      <div class="col-md-8 col-lg-6 col-xl-4 offset-xl-1">

        <form class="mx-1 mx-md-4" action="{{ url_for('login') }}" method="post">

          <div class="d-flex align-items-center mb-3 pb-1">

            <span class="h1 fw-bold mb-0">Login</span>

          </div>

          <!-- Email input -->

          <div class="form-outline mb-4">

            <input type="email" name="email" value="" placeholder="Email ID" class="form-control" />

          </div>

          <!-- Password input -->

          <div class="form-outline mb-3">

            <input type="password" name="password" value="" placeholder="Password" class="form-control" />

          </div>

          <div class="text-center text-lg-start mt-4 pt-2">

            <input type="submit" class="btn btn-primary btn-lg" style="width: 125px; display: block; margin-left:
auto; margin-right: auto;">

          </div>

          <p class="support-p" style="margin-top: 10px;">Don't have an Account?

            <a href="/register" style="color: rgb(0, 0, 0); text-decoration: none;"><strong style="font-size:
large;">Register</strong></a>

          </p>

        </form>

```

```
</div>
</div>
</div>
</section>
{% endblock %}
```

mystyle.css

```
.
r
i
g
h
t
{
d
i
s
p
l
a
y
:
b
l
o
c
k
;
```



```
f  
l  
o  
a  
t  
:  
r  
i  
g  
h  
t  
;  
c  
o  
l  
o  
r  
:  
b  
l  
u  
e  
;  
}  
.bg { background-image: url('pexels-  
francesco-ungaro281260.jpg');  
background-position: center;
```

```
backgroundrepeat: no-repeat;
background-size: cover;
}
```

DASHBOARD

Users.html

```
{% extends 'dashboard/base.html' %}
```

```
{% block head %}
```

```
<title>Dashboard</title>
```

```
<style>
```

```
.container
```

```
{
```

```
paddin
```

```
g:
```

```
20px;
```

```
border
```

```
: 2px
```

```
5px
```

```
5px
```

```
5px;
```

```
border
```

```
-color:
```

```
#ffffff;
```

```
border
```

```
-
```

```
width:
```

```
5px;
```

```
border
```

```
-style:
```

```
solid;
textali
gn:
center
;
border
radius:
15px;
boxsh
adow:
1px
1px
10px
5px
rgba(2
34,
234,
234,
0.844);
margin
-
botto
m:
10px;
margin
-top:
10px;
margin
```

```

-left:
10px;
margin
-right:
15px;
backgr
ound-
color:
#ffffff;
transit
ion:
transf
orm
1.0s
ease }
.contai
ner:ho
ver {
transf
orm:
scale(1
.05); }
</style>
{% endblock %}
{% block body %}
<div class="home-content">
  <div class="overview-boxes">
    <section class="catogories" >

```

```

<div class="container-fluid"> <div class="row">

<div class="col-lg-6">

  <div class="row">

    <div class="col-lg-6 col-md-6 col-12 p-1">

      <div class="catagories_item">

        <div class="container">

          <a href="/productlist" style="text-decoration: none; color: black">

            <div class="box-topic">Products List

            </div></a>

          <a href="/productlist" style="text-decoration: none; color: black">

            </a>

          </div>

        </div>

      </div>

    <div class="col-lg-6 col-md-6 col-12 p-1">

      <div class="catagories_item">

        <div class="container">

          <a href="/addproduct" style="text-decoration: none; color: black">

            <div class="box-topic">Add Products

            </div></a>

          <a href="/addproduct" style="text-decoration: none; color: black">

            </a>

          </div>

        </div>

      </div>

    </div>

  </div>

</div>

```

```

<div class="col-lg-6">
  <div class="row">
    <div class="col-lg-6 col-md-6 col-12 p-1">
      <div class="catagories_item">
        <div class="container">
          <a href="/movement" style="text-decoration: none; color: black">
            <div class="box-topic">Products Movement
          </div></a>
          <a href="/movement" style="text-decoration: none; color: black">
            </a>
          </div>
        </div>
      </div>
    <div class="col-lg-6 col-md-6 col-12 p-1">
      <div class="catagories_item">
        <div class="container">
          <a href="/report" style="text-decoration: none; color: black">
            <div class="box-topic">Product Report
          </div></a>
          <a href="/report" style="text-decoration: none; color: black">
            
            </a>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>

```

```

        </div>
    </div>
</div>
</section>
</div>
</div>
{{msg}}
{% endblock %}

```

special.html

```

{% extends 'dashboard/base.html' %}

{% block head %}

    <title>List of Users</title>

{% endblock %}

{% block body %}

<link href="//maxcdn.bootstrapcdn.com/bootstrap/3.3.0/css/bootstrap.min.css" rel="stylesheet" id="bootstrap-
css">

<div > </div>

    <div class="home-content">

        <div class="overview-boxes">

            <section
class="catogori
es" >    <div
class="contain
er-fluid">

                <div class="row">

                    <div class="col-lg-6">

                        <div class="row">

                            <div class="col-lg-6 col-md-6 col-12 p-1">

```

```

        <div class="catagories_item">
    </div>
</div>
<div class="col-lg-6 p-0" style=" margin-top: 25px;">
    <div class="catagories_item">
        <div class="catagories_item catagories_large_item" >
            </div>
        </div>
    </div>
</div>
<div class="col-lg-6 col-md-6 col-12 p-1">
    <div class="catagories_item">
        </div>
    </div>
<div class="col-lg-6 col-md-6 col-12 p-1">
    <div class="catagories_item">

        </div>
    </div>
</div>
</div>
<div class="container register-form">
    <div class="form">
        <div class="d-flex align-items-center mb-3 pb-1">
            {{msg}}
            <span class="h1 fw-bold mb-0">Product List</span>
        </div>
        <table class="table table-hover">
            <thead>
                <tr>

```



```

7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32733;SECURITY=SSL;SSLServerCertificate
=DigiCertGlobalRootCA.crt;UID=lkc93724;PWD=zAzNGa6DaNk6xvle",",") import smtplib, ssl ## email.mime
subclasses from email.mime.multipart import MIMEMultipart from email.mime.text import MIMEText
## The pandas library is only for generating the current date, which is not necessary for
sending emails import pandas as pd app = Flask(__name__)

var
_li
st
=
[]
ap
p.s
ecr
et
_k
ey
=
'yo
ur
se
cre
t
ke
y'
@
ap
p.r
ou

```

```

te(
'/')
de
f
ho
m
e()
:
if
no
t
ses
sio
n.g
et(
"n
am
e")
:
    return render_template('home.html') return
render_template('home.html', session = session)
@app.route('/re
gister') def
new_student():
    return render_template('Register.html')
@app.route('/addrec',methods =
['POST', 'GET']) def addrec(): if
request.method == 'POST':

```

```

fname = request.form['fname']
lname = request.form['lname']
cname = request.form['cname']
state = request.form['state'] city
= request.form['city'] mobileno
= request.form['mobileno']
emailid = request.form['emailid']
password =
request.form['password']
pincode = request.form['pincode']
sql = "SELECT * FROM Users
WHERE EMAILID =?"
stmt =
ibm_db.prepare(conn,
sql)
ibm_db.bind_param(s
tmt,1,emailid)
ibm_db.execute(stmt)
account =
ibm_db.fetch_assoc(st
mt) if account: users
= [] sql = "SELECT *
FROM Users" stmt
=
ibm_db.exec_immedia
te(conn, sql)
dictionary =
ibm_db.fetch_both(st

```

```

mt)    while dictionary
!= False:
        # print ("The Name is : ", dictionary)    users.append(dictionary)    dictionary =
ibm_db.fetch_both(stmt)    return render_template('list.html', msg="You are already a member,
please login using your details", users = users)    else:
        var_list.append(fname)
var_list.append(lname)
var_list.append(cname)
var_list.append(state)
var_list.append(city)
var_list.append(mobileno)
var_list.append(emailid)
var_list.append(password)
var_list.append(pincod)

bodytemp = r'D:\IBM\GUIDED PROJECT\INVENTORY MANAGEMENT SYSTEM FOR
RETAILERS\templates\email.html'    with open(bodytemp, "r", encoding='utf-8') as f:    html= f.read()

# Set up the email addresses and password. Please replace below with your email address and
password    email_from = 'padhu10a@gmail.com'    epassword = 'rbjibzkssszsbrjo'    email_to =
emailid

# Generate today's date to be included in the
email Subject    date_str =
pd.Timestamp.today().strftime('%Y-%m-%d')

# Create a MIMEMultipart class, and set up the From, To, Subject
fields    email_message = MIMEMultipart()
email_message['From'] = email_from    email_message['To'] =
email_to    email_message['Subject'] = f'Report email - {date_str}'

# Attach the html doc defined earlier, as a MIMEText html content type to the MIME message
email_message.attach(MIMEText(html, "html"))

```

```

    # Convert it as a string    email_string =
email_message.as_string() # Connect to the Gmail SMTP
server and Send Email    context =
ssl.create_default_context()    with
smtplib.SMTP_SSL("smtp.gmail.com", 465,
context=context) as server:
    server.login(email_from, epassword)
server.sendmail(email_from, email_to, email_string)    return
render_template('notify.html')

@app.route('/c
onfirm') def
confirmation():
    insert_sql = "INSERT INTO Users (FIRSTNAME, LASTNAME, COMPANYNAME, STATE, CITY, MOBILENO, EMAILID,
PASSWORD, PINCODE) VALUES (?, ?, ?, ?, ?, ?, ?, ?)"    prep_stmt = ibm_db.prepare(conn, insert_sql)
ibm_db.bind_param(prepare_stmt, 1, var_list[0])    ibm_db.bind_param(prepare_stmt, 2, var_list[1])
ibm_db.bind_param(prepare_stmt, 3, var_list[2])    ibm_db.bind_param(prepare_stmt, 4, var_list[3])
ibm_db.bind_param(prepare_stmt, 5, var_list[4])
ibm_db.bind_param(prepare_stmt, 6, var_list[5])    ibm_db.bind_param(prepare_stmt, 7, var_list[6])
ibm_db.bind_param(prepare_stmt, 8, var_list[7])    ibm_db.bind_param(prepare_stmt, 9, var_list[8])
ibm_db.execute(prepare_stmt)    return render_template('confirm.html')

@app.route('/login', methods=['POST', 'GET']) def login():    msg = "

if request.method == 'POST' and 'email' in request.form and

'password' in request.form:

    email = request.form['email']
password = request.form['password']

    sql = "SELECT * FROM Users WHERE EMAILID =? AND
PASSWORD =?"    stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt,1,email)

```

```

ibm_db.bind_param(stmt,2,password)
ibm_db.execute(stmt)    account =
ibm_db.fetch_assoc(stmt) if account:
    session['loggedin'] = True    session['id'] =
account['ID']    session['email']
= account['EMAILID']
session['name'] =
account['FIRSTNAME']    msg =
'Logged in successfully !'
return
render_template('ho
me.html', msg = msg)
else:
    msg = 'Incorrect
email / password !'
return
render_template('login.ht
ml', msg = msg)

```

@

a

p

p

.

r

o

u

t

e
(
,
/
l
o
g
o
u
t
,
)
d
e
f
l
o
g
o
u
t
(
)
:

session.p
op('logge
din',


```
None)
session.p
op('id',
None)
session.p
op('email
', None)
session.p
op('name
', None)
return
redirect(
url_for('home'))
@app.route('/list')
def list():
    users = []
    sql = "SELECT *
FROM Users"
    stmt =
    ibm_db.exec_immediate(
conn, sql)
    dictionary =
    ibm_db.fetch_both(
stmt)
    while
dictionary != False:
```

```

# print ("The Name
is : ", dictionary)

users.append(dictio
nary)  dictionary =
ibm_db.fetch_both(
stmt) if users:

    return render_template("list.html", users = users , session = session) return

"No users..."

```

Users.html

```

<html>

<head>

<div class="row">

    <div class="col-md-12">

        </div>

    </div>

<div class="row">

    <div class="col-md-12">

        <div class="panel panel-default">

            <div class="panel-heading clearfix">

                <strong>

                    <span class="glyphicon glyphicon-th"></span>

                    <span>Users</span>

                </strong>

                <a href="add_user. class="btn btn-info pull-right">Add New User</a>

            </div>

            <div class="panel-body">

```

```

<table class="table table-bordered table-striped">
  <thead>
    <tr>
      <th class="text-center" style="width: 50px;">#</th>
      <th>Name </th>
      <th>Username</th>
      <th class="text-center" style="width: 15%;">User Role</th>
      <th class="text-center" style="width: 10%;">Status</th>
      <th style="width: 20%;">Last Login</th>
      <th class="text-center" style="width: 100px;">Actions</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td class="text-center"> </td>
      <td class="text-center">
        <td class="text-center">
          <span class="label label-success">
            </td>
            <td>
              <td class="text-center">
                <div class="btn-group">
                  <a href="edit_user.id=" class="btn btn-xs btn-warning" data-toggle="tooltip" title="Edit">
                    <i class="glyphicon glyphicon-pencil"></i>
                  </a>
                  <i class="glyphicon glyphicon-remove"></i>
                </a>
              </div>
            </td>

```



```

<thead>

<tr>

  <th class="text-center" style="width: 50px;">#</th>

  <th>Group Name</th>

  <th class="text-center" style="width: 20%;">Group Level</th>

  <th class="text-center" style="width: 15%;">Status</th>

  <th class="text-center" style="width: 100px;">Actions</th>

</tr>

</thead>

<tbody>

<tr>

  <td class="text-center"></td>

  <td class="text-center">

</td>

  <td class="text-center">

    <span class="label label-success"><span>

<span class="label label-danger"></span>

</td>

  <td class="text-center">

    <div class="btn-group">

      <a href="edit_group.html id=label.html class="btn btn-xs btn-warning" datatoggle="tooltip" title="Edit">

        <i class="glyphicon glyphicon-pencil"></i>

      </a>

      <a href="delete_group.html id=label.html" class="btn btn-xs btn-danger" datatoggle="tooltip"
title="Remove">

        <i class="glyphicon glyphicon-remove"></i>

      </a>

    </div>

  </td>

```

```

        </tr>
    </tbody>
</table>

</div>

</div>

</div></div>

```

newgroup.html

```

{% extends 'base.html' %}

{% block head %}

    <title>List of Users</title>

{% endblock %}

{% block body %}

<!-- Hero Area Start-->

    {{ msg }}

    {{ session['name'] }}

    <table border = 1>

        <thead>

            <td>FISRT NAME</td>

            <td>LAST NAME</td>

            <td>COMPANY NAME</td>

            <td>STATE</td>

            <td>CITY</td>

            <td>MOBILENO</td>

            <td>EMAILID</td>

            <td>PASSWORD</td>

            <td>PINCODE</td>

        </thead>

        {% for row in users %}

            <tr>

```

```

        <td>{{row["FIRSTNAME"]}}</td>        <td> {{ row["LASTNAME"]}}</td>
        <td>{{row["COMPANYNAME"]}}</td>
        <td>{{row['STATE']}}</td>
        <td>{{row["CITY"]}}</td>
        <td> {{ row["MOBILENO"]}}</td>
        <td>{{row["EMAILID"]}}</td>
        <td>{{row['PASSWORD']}}</td>
        <td>{{row['PINCODE']}}</td>
    </tr>
    {% endfor %}
</table>
{% endblock %}

```

newuser.html

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Transitional //EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xmlns:v="urn:schemas-microsoft-com:vml" xmlns:o="urn:schemas-
microsoft-com:office:office">

<head>

    <!--[if gte mso 9]>

<xml>

    <o:OfficeDocumentSettings>

        <o:AllowPNG/>

        <o:PixelsPerInch>96</o:PixelsPerInch>

    </o:OfficeDocumentSettings>

</xml>

<![endif]-->

    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<meta name="x-apple-disable-message-reformatting">
<!--[if !mso]><!-->
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<!--<![endif]>
<title></title>
<style type="text/css">
    @media only screen and (min-width: 620px) {
        .u-row {      width: 600px !important;
        }
        .u-row .u-col {      vertical-align: top;
        }
        .u-row .u-col-100 {      width: 600px !important;
        }
    }

    @media (max-width: 620px) {      .u-row-container {
max-width: 100% !important;      padding-left: 0px
!important;      padding-right: 0px !important;
    }
        .u-row .u-col {      min-width: 320px !important;
max-width: 100% !important;      display: block
!important;
        }
        .u-row {      width: calc(100% - 40px) !important;      }
        .u-col {      width: 100% !important;
        }
        .u-col>div {      margin: 0 auto;
        }
    }

```



```

body {
    margin: 0;    padding: 0;
} table, tr, td {    vertical-align: top;
border-collapse: collapse;
} p {    margin: 0;
}

.ie-container table, .mso-container table {
table-layout: fixed;
} * {    line-height: inherit;
}
a[x-apple-data-detectors='true'] {    color: inherit !important;    text-decoration: none !important;
} table, td {    color: #000000;
}

#u_body a {    color: #0000ee; text-
decoration: underline;
}
</style>
<!--[if !mso]><!-->
<link href="https://fonts.googleapis.com/css?family=Cabin:400,700" rel="stylesheet" type="text/css">
<!--<![endif]-->
</head>
<body class="clean-body u_body" style="margin: 0;padding: 0;-webkit-text-size-adjust:
100%;background-color: #f9f9f9;color: #000000">
<!--[if IE]><div class="ie-container"><![endif]-->
<!--[if mso]><div class="mso-container"><![endif]-->

<table id="u_body" style="border-collapse: collapse;table-layout: fixed;border-spacing: 0;mso-table-lspace:
0pt;mso-table-rspace: 0pt;vertical-align: top;min-width: 320px;Margin: 0 auto;background-color:
#f9f9f9;width:100%" cellpadding="0" cellspacing="0">

<tbody>

```

```

<tr style="vertical-align: top">

  <td style="word-break: break-word;border-collapse: collapse !important;vertical-align: top">

    <!--[if (mso)|(IE)]><table width="100%" cellpadding="0" cellspacing="0" border="0"><tr><td align="center"
style="background-color: #f9f9f9;"><![endif]-->

    <div class="u-row-container" style="padding: 0px;background-color: transparent">

      <div class="u-row" style="Margin: 0 auto;min-width: 320px;max-width: 600px;overflow-wrap: break-
word;word-wrap: break-word;word-break: breakword;background-color: transparent;">

        <div style="border-collapse: collapse;display: table;width: 100%;height:
100%;background-color: transparent;">

          <!--[if (mso)|(IE)]><table width="100%" cellpadding="0" cellspacing="0" border="0"><tr><td
style="padding: 0px;background-color: transparent;" align="center"><table cellpadding="0" cellspacing="0"
border="0" style="width:600px;"><tr style="background-color: transparent;"><![endif]-->

          <!--[if (mso)|(IE)]><td align="center" width="600" style="width: 600px;padding: 0px;border-top: 0px solid
transparent;border-left: 0px solid transparent;border-right: 0px solid transparent;border-bottom: 0px solid
transparent;" valign="top"><![endif]-->

            <div class="u-col u-col-100" style="max-width: 320px;min-width: 600px;display: table-cell;vertical-align:
top;">

              <div style="height: 100%;width: 100% !important;">

                <!--[if (!mso)&(!IE)]><!-->

                <div style="height: 100%; padding: 0px;border-top: 0px solid transparent;border-left: 0px solid
transparent;border-right: 0px solid transparent;borderbottom: 0px solid transparent;">

                  <!--<![endif]-->

                <table style="font-family:'Cabin',sans-serif;" role="presentation" cellpadding="0" cellspacing="0" width="100%"
border="0">

                  <tbody>

                    <tr>

                      <td style="overflow-wrap:break-word;word-break:breakword;padding:10px;font-family:'Cabin',sans-
serif;" align="left">

                        <div style="color: #afb0c7; line-height: 170%; text-align: center; word-wrap: break-word;">

                          <p style="font-size: 14px; line-height: 170%;"><span style="font-size: 14px; line-height:
23.8px;">View Email in Browser</span></p>

                        </div>

                      </td>

```

```

        </tr>
    </tbody>
</table>
<!--[if (!mso)&(!IE)]><!-->
    </div>
    <!--<![endif]-->
</div>
</div>
</div>
<!--[if (mso)|(IE)]></td><![endif]-->
<!--[if (mso)|(IE)]></tr></table></td></tr></table><![endif]-->
</div>
</div>
</div>
<div class="u-row-container" style="padding: 0px;background-color: transparent">
    <div class="u-row" style="Margin: 0 auto;min-width: 320px;max-width: 600px;overflow-wrap: break-
word;word-wrap: break-word;word-break: breakword;background-color: #ffffff;">
        <div style="border-collapse: collapse;display: table;width: 100%;height:
100%;background-color: transparent;">
            <!--[if (mso)|(IE)]><table width="100%" cellpadding="0" cellspacing="0" border="0"><tr><td
style="padding: 0px;background-color: transparent;" align="center"><table cellpadding="0" cellspacing="0"
border="0" style="width:600px;"><tr style="background-color: #ffffff;"><![endif]-->
            <!--[if (mso)|(IE)]><td align="center" width="600" style="width: 600px;padding: 0px;border-top: 0px solid
transparent;border-left: 0px solid transparent;border-right: 0px solid transparent;border-bottom: 0px solid
transparent;" valign="top"><![endif]-->
            <div class="u-col u-col-100" style="max-width: 320px;min-width: 600px;display: table-cell;vertical-align:
top;">
                <div style="height: 100%;width: 100% !important;">
                    <!--[if (!mso)&(!IE)]><!-->
                    <div style="height: 100%; padding: 0px;border-top: 0px solid transparent;border-left: 0px solid
transparent;border-right: 0px solid transparent;borderbottom: 0px solid transparent;">
                        <!--<![endif]-->

```

```

<table style="font-family:'Cabin',sans-serif;" role="presentation" cellpadding="0" cellspacing="0" width="100%"
border="0">

    <tbody>

        <tr>

            <td style="overflow-wrap:break-word;word-break:breakword;padding:20px;font-family:'Cabin',sans-
serif;" align="left">

<table width="100%" cellpadding="0" cellspacing="0" border="0">

    <tr>

<td style="padding-right: 0px;padding-left: 0px;" align="center">



    </td>

        </tr>

    </table>

    </td>

</tr>

</tbody>

</table>

<!--[if (!mso)&(!IE)]><!-->

</div>

<!--<![endif]-->

</div>

</div>

<!--[if (mso)|(IE)]></td><![endif]-->

<!--[if (mso)|(IE)]></tr></table></td></tr></table><![endif]-->

</div>

</div>

```

```

</div>

<div class="u-row-container" style="padding: 0px;background-color: transparent">

    <div class="u-row" style="Margin: 0 auto;min-width: 320px;max-width: 600px;overflow-wrap: break-
word;word-wrap: break-word;word-break: breakword;background-color: #003399;">

        <div style="border-collapse: collapse;display: table;width: 100%;height:
100%;background-color: transparent;">

            <!--[if (mso)|(IE)]><table width="100%" cellpadding="0" cellspacing="0" border="0"><tr><td
style="padding: 0px;background-color: transparent;" align="center"><table cellpadding="0" cellspacing="0"
border="0" style="width:600px;"><tr style="background-color: #003399;"><![endif]-->

            <!--[if (mso)|(IE)]><td align="center" width="600" style="width: 600px;padding: 0px;border-top: 0px solid
transparent;border-left: 0px solid transparent;border-right: 0px solid transparent;border-bottom: 0px solid
transparent;" valign="top"><![endif]-->

                <div class="u-col u-col-100" style="max-width: 320px;min-width: 600px;display: table-cell;vertical-align:
top;">

                    <div style="height: 100%;width: 100% !important;">

                        <!--[if (!mso)&(!IE)]><!-->

                            <div style="height: 100%; padding: 0px;border-top: 0px solid transparent;border-left: 0px solid
transparent;border-right: 0px solid transparent;borderbottom: 0px solid transparent;">

                                <!--<![endif]-->

                                <table style="font-family:'Cabin',sans-serif;" role="presentation" cellpadding="0" cellspacing="0" width="100%"
border="0">

                                    <tbody>

                                        <tr>

                                            <td style="overflow-wrap:break-word;word-break:breakword;padding:40px 10px 10px;font-
family:'Cabin',sans-serif;" align="left">

                                                <table width="100%" cellpadding="0" cellspacing="0" border="0">

                                                    <tr>

                                                        <td style="padding-right: 0px;padding-left: 0px;" align="center">

                                                        </td>

```

```

        </tr>
    </table>

    </td>

</tr>

</tbody>

</table>

<table style="font-family:'Cabin',sans-serif;" role="presentation" cellpadding="0" cellspacing="0"
width="100%" border="0">

    <tbody>

        <tr>

            <td style="overflow-wrap:break-word;word-break:breakword;padding:10px;font-family:'Cabin',sans-
serif;" align="left">

                <div style="color: #e5eaf5; line-height: 140%; text-align: center; word-wrap: break-word;">
<p style="font-size: 14px; line-height: 140%;"><strong>T H A N K S   F O R   S I G N I N G   U P !</strong></p>

                </div>

            </td>

        </tr>

    </tbody>

</table>

<table style="font-family:'Cabin',sans-serif;" role="presentation" cellpadding="0" cellspacing="0" width="100%"
border="0">

    <tbody>

        <tr>

            <td style="overflow-wrap:break-word;word-break:break-word;padding:0px
10px 31px;font-family:'Cabin',sans-serif;" align="left">

                <div style="color: #e5eaf5; line-height: 140%; text-align: center; word-wrap: break-word;">

                    <p style="font-size: 14px; line-height: 140%;"><span style="font-size:
28px; line-height: 39.2px;"><strong><span style="line-height: 39.2px; font-size: 28px;">Confirm Your E-mail Address
</span></strong>

                    </span>

                </p>

            </td>

        </tr>

    </tbody>

</table>

```

```

        </div>
    </td>
</tr>
</tbody>
</table>
<!--[if (!mso)&(!IE)]><!-->
</div>
<!--<![endif]-->
</div>
</div>
<!--[if (mso)|(IE)]></td><![endif]-->
<!--[if (mso)|(IE)]></tr></table></td></tr></table><![endif]-->
</div>
</div>
</div>
<div class="u-row-container" style="padding: 0px;background-color: transparent">
    <div class="u-row" style="Margin: 0 auto;min-width: 320px;max-width: 600px;overflow-wrap: break-
word;word-wrap: break-word;word-break: breakword;background-color: #ffffff;">
        <div style="border-collapse: collapse;display: table;width: 100%;height:
100%;background-color: transparent;">
            <!--[if (mso)|(IE)]><table width="100%" cellpadding="0" cellspacing="0" border="0"><tr><td
style="padding: 0px;background-color: transparent;" align="center"><table cellpadding="0" cellspacing="0"
border="0" style="width:600px;"><tr style="background-color: #ffffff;"><![endif]-->
<!--[if (mso)|(IE)]><td align="center" width="600" style="width: 600px;padding: 0px;border-top: 0px solid
transparent;border-left: 0px solid transparent;border-right: 0px solid transparent;border-bottom: 0px solid
transparent;" valign="top"><![endif]-->
            <div class="u-col u-col-100" style="max-width: 320px;min-width: 600px;display: table-cell;vertical-align:
top;">
                <div style="height: 100%;width: 100% !important;">
                    <!--[if (!mso)&(!IE)]><!-->
                    <div style="height: 100%; padding: 0px;border-top: 0px solid transparent;border-left: 0px solid
transparent;border-right: 0px solid transparent;borderbottom: 0px solid transparent;">

```

<!--<![endif]-->

<table style="font-family:'Cabin',sans-serif;" role="presentation" cellpadding="0" cellspacing="0" width="100%" border="0">

<tbody>

<tr>

<td style="overflow-wrap:break-word;word-break:breakword;padding:33px 55px;font-family:'Cabin',sans-serif;" align="left">

<div style="line-height: 160%; text-align: center; word-wrap: break-word;">

<p style="font-size: 14px; line-height: 160%;">Hi, </p>

<p style="font-size: 14px; line-height: 160%;">You're almost ready to get started. Please click on the button below to confirm your email address and experience the awesome Inventory Management Service!</p>

</div>

</td>

</tr>

</tbody>

</table>

<table style="font-family:'Cabin',sans-serif;" role="presentation" cellpadding="0" cellspacing="0" width="100%" border="0">

<tbody>

<tr>

<td style="overflow-wrap:break-word;word-break:breakword;padding:10px;font-family:'Cabin',sans-serif;" align="left">

<!--[if mso]><style>.v-button {background: transparent !important;}</style><![endif]-->

<div align="center">

<!--[if mso]><v:roundrect xmlns:v="urn:schemas-microsoft-com:vml" xmlns:w="urn:schemas-microsoft-com:office:word" href="http://localhost:5000/confirm" style="height:46px; v-text-anchor:middle; width:157px;" arcsize="8.5%" stroke="f" fillcolor="#9ecbe4"><w:anchorlock/><center style="color:#103c55;font-family:'Cabin',sans-serif;"><![endif]-->

<a href="http://localhost:5000/confirm" target="_blank" class="v-button" style="boxsizing: border-box;display: inline-block;font-family:'Cabin',sans-serif;text-decoration: none;webkit-text-size-adjust: none;text-align:

center;color: #103c55; background-color: #9ecbe4; border-radius: 4px;-webkit-border-radius: 4px; -moz-border-radius: 4px; width:auto; maxwidth:100%; overflow-wrap: break-word; word-break: break-word; word-wrap:break-word; mso-border-alt: none;">

```
<span style="display:block;padding:14px 44px 13px;lineheight:120%;"><span style="font-size:
16px; line-height: 19.2px;"><strong><span style="line-height: 19.2px; font-size: 16px;">CONFIRM</span></strong>
</span>
</span>
</a>
<!--[if mso]></center></v:roundrect><![endif]-->
</div>
</td>
</tr>
</tbody>
</table>
```

<table style="font-family:'Cabin',sans-serif;" role="presentation" cellpadding="0" cellspacing="0" width="100%" border="0">

```
<tbody>
<tr>
<td style="overflow-wrap:break-word;word-break:breakword;padding:33px 55px 60px;font-
family:'Cabin',sans-serif;" align="left">
<div style="color: #3598db; line-height: 160%; text-align: center; wordwrap: break-word;">
<p style="line-height: 160%; font-size: 14px;"><span style="font-size:
18px; line-height: 28.8px;">Once again, Thanks for signing up with us!</span></p>
</div>
</td>
</tr>
</tbody>
</table>
<!--[if (!mso)&(!IE)]><!-->
</div>
```

```

        <!--<![endif]-->

    </div>

</div>

<!--[if (mso)|(IE)]></td><![endif]-->

<!--[if (mso)|(IE)]></tr></table></td></tr></table><![endif]-->

</div>

</div>

</div>

<!--[if (mso)|(IE)]></td></tr></table><![endif]-->

</td>

</tr>

</tbody>

</table>

<!--[if mso]></div><![endif]-->

<!--[if IE]></div><![endif]-->

</body>

</html>

```

mystyle.css

```

.right { display: block; float: right;

color: blue;

}

.bg { background-image: url('pexels-francesco-ungaro-281260.jpg'); background-
position: center; background-repeat: no-repeat; background-size: cover;

}

```

app.py

```

from turtle import st from flask import Flask, render_template, request, redirect, url_for, session

from markupsafe import escape import ibm_db

```

```

conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=54a2f15b-5c0f-46df-
89547e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32733;SECURITY
=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=lkc93724;PWD=zAzNGa6Da
Nk6xvle",",") import smtplib, ssl ## email.mime subclasses from

email.mime.multipart import MIMEMultipart from email.mime.text

import MIMEText

## The pandas library is only for generating the current date, which is not necessary for sending emails import
pandas as pd from datetime import datetime from flask import Flask app = Flask(__name__)

var_list = [] app.secret_key = 'your secret key'

@app.route('/') def home():      if not
session.get("name"):

    return render_template('home.html') return
render_template('home.html', session = session)

@app.route('/register') def new_student():

    return render_template('Register.html') @app.route('/addrec',methods =

['POST', 'GET']) def addrec():

    if request.method == 'POST':    fname =
request.form['fname']    lname = request.form['lname']

cname = request.form['cname']    state =
request.form['state']    city = request.form['city']    mobileno

= request.form['mobileno']    emailid =
request.form['emailid']    password =
request.form['password']    pincode =
request.form['pincode']

    sql = "SELECT * FROM Users WHERE EMAILID =?"

    stmt = ibm_db.prepare(conn, sql)

ibm_db.bind_param(stmt,1,emailid)

ibm_db.execute(stmt)    account =

ibm_db.fetch_assoc(stmt) if account:    users = []

```

```

    sql = "SELECT * FROM Users"    stmt =
ibm_db.exec_immediate(conn, sql)    dictionary =
ibm_db.fetch_both(stmt)    while dictionary != False:
    #    print    ("The    Name    is    :    ",    dictionary)
users.append(dictionary)    dictionary =
ibm_db.fetch_both(stmt)

    return render_template('list.html', msg="You are already a member, please login using your details", users =
users)

else:
    var_list.append(fname)    var_list.append(lname)
var_list.append(cname)    var_list.append(state)
var_list.append(city)    var_list.append(mobileno)
var_list.append(emailid)    var_list.append(password)
var_list.append(pincode)

    bodytemp = r"D:\IBM\GUIDED PROJECT\INVENTORY MANAGEMENT SYSTEM FOR RETAILERS\SPRINT
2\templates\email.html"

    with open(bodytemp, "r", encoding='utf-8') as f:
        html= f.read()

    # Set up the email addresses and password. Please replace below with your email address and password
email_from = 'padhu10a@gmail.com'    epassword = 'rbjibzkssszsbrjo'    email_to = emailid

    # Generate today's date to be included in the email Subject    date_str =
pd.Timestamp.today().strftime('%Y-%m-%d')    # Create a MIMEMultipart class, and set up
the From, To, Subject fields    email_message = MIMEMultipart()
email_message['From'] = email_from    email_message['To'] = email_to
email_message['Subject'] = f'Report email - {date_str}'

    # Attach the html doc defined earlier, as a MIMEText html content type to the MIME message
email_message.attach(MIMEText(html, "html"))

```

```

    # Convert it as a string    email_string = email_message.as_string() # Connect to the Gmail SMTP
server and Send Email    context = ssl.create_default_context()    with
smtpplib.SMTP_SSL("smtp.gmail.com", 465, context=context) as server:

    server.login(email_from, epassword)    server.sendmail(email_from,
email_to, email_string)    return render_template('notify.html')

@app.route('/confirm') def confirmation():

    insert_sql = "INSERT INTO Users (FIRSTNAME, LASTNAME, COMPANYNAME,
STATE, CITY, MOBILENO, EMAILID, PASSWORD, PINCODE) VALUES
(?,?,?,?,?,?,?,?)"    prep_stmt = ibm_db.prepare(conn, insert_sql)

    ibm_db.bind_param(prepare_stmt, 1, var_list[0])
    ibm_db.bind_param(prepare_stmt, 2, var_list[1])
    ibm_db.bind_param(prepare_stmt, 3, var_list[2])
    ibm_db.bind_param(prepare_stmt, 4, var_list[3])
    ibm_db.bind_param(prepare_stmt, 5, var_list[4])
    ibm_db.bind_param(prepare_stmt, 6, var_list[5])
    ibm_db.bind_param(prepare_stmt, 7, var_list[6])
    ibm_db.bind_param(prepare_stmt, 8, var_list[7])
    ibm_db.bind_param(prepare_stmt, 9, var_list[8])

    ibm_db.execute(prepare_stmt)    return render_template('confirm.html')

@app.route('/login', methods=['POST', 'GET']) def login():    msg = "
if request.method == 'POST' and 'email' in request.form and
'password' in request.form:

    email = request.form['email']    password =
request.form['password']

    sql = "SELECT * FROM Users WHERE EMAILID =? AND PASSWORD =?"

    stmt    =    ibm_db.prepare(conn,    sql)

    ibm_db.bind_param(stmt,1,email)

    ibm_db.bind_param(stmt,2,password)    ibm_db.execute(stmt)

    account = ibm_db.fetch_assoc(stmt) if account:

```

```

        session['loggedin'] = True        session['id'] = account['ID']        session['email'] =
account['EMAILID']        session['name'] = account['FIRSTNAME']        msg = 'Logged in
successfully !'        return render_template('dashboard/dashboard.html', msg = msg)

    else:

        msg = 'Incorrect email / password !'    return
render_template('login.html', msg = msg)

@app.route('/dashboard') def dashboard():

    return render_template('dashboard/dashboard.html')

@app.route('/addproduct')

def addproduct():

    return render_template('dashboard/addproduct.html')

@app.route('/movements') def movement():

products = []

    sql = "SELECT * FROM Products WHERE HOLDERNAME = ?"

    prep_stmt = ibm_db.prepare(conn, sql)

    ibm_db.bind_param(prepare_stmt, 1, session['name'])

    ibm_db.execute(prepare_stmt)    dictionary = ibm_db.fetch_both(prepare_stmt)

    while dictionary != False:

        # print ("The Name is : ", dictionary)

    products.append(dictionary)    dictionary =

    ibm_db.fetch_both(prepare_stmt)    if products:

        return render_template("dashboard/movement.html", products = products , session = session)

@app.route('/moveproc') def moveproc():

    return render_template('dashboard/dashboard.html')

@app.route('/report') def report():

    return render_template('dashboard/report.html')

@app.route('/stockupdate') def stock():

    return render_template('dashboard/stockupdate.html')

```

```

@app.route('/index') def index():

    return render_template('dashboard/index.html') @app.route('/addproc',methods = ['POST',
'GET']) def addproc():    if request.method == 'POST':        pname = request.form['pname']
quantity = request.form['quantity']        the_time = datetime.now()        the_time =
the_time.replace(second=0, microsecond=0)        sql = "SELECT * FROM Products WHERE
HOLDERNAME =?"

        stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt,1,session['name'])
ibm_db.execute(stmt)        product = ibm_db.fetch_assoc(stmt)

        if product:

            if product['PRODUCTNAME']==pname:

                return render_template('dashboard/addproduct.html', msg="Product already added! Add a new
product.")

            else:

                sql ="INSERT INTO Products
(PRODUCTNAME,QUANTITY,FROM,TO,DATE,HOLDERNAME) VALUES
(?,?,?, ?,?);"        prep_stmt = ibm_db.prepare(conn, sql)        ibm_db.bind_param(prepare_stmt, 1,
pname)        ibm_db.bind_param(prepare_stmt, 2, quantity)        ibm_db.bind_param(prepare_stmt, 3, "")
ibm_db.bind_param(prepare_stmt, 4, "")        ibm_db.bind_param(prepare_stmt, 5, str(the_time))
ibm_db.bind_param(prepare_stmt, 6, session['name'])        ibm_db.execute(prepare_stmt)        return
render_template('dashboard/addproduct.html', msg="Product added")        else:

                sql ="INSERT INTO Products
(PRODUCTNAME,QUANTITY,FROM,TO,DATE,HOLDERNAME) VALUES
(?,?,?, ?,?);"        prep_stmt = ibm_db.prepare(conn, sql)        ibm_db.bind_param(prepare_stmt, 1,
pname)        ibm_db.bind_param(prepare_stmt, 2, quantity)        ibm_db.bind_param(prepare_stmt, 3, "")
ibm_db.bind_param(prepare_stmt, 4, "")        ibm_db.bind_param(prepare_stmt, 5, str(the_time))
ibm_db.bind_param(prepare_stmt, 6, session['name'])        ibm_db.execute(prepare_stmt)        return
render_template('dashboard/addproduct.html', msg="Product added")

```

```

@app.route('/productlist') def productlist():
    products = []

    sql = "SELECT * FROM Products WHERE HOLDERNAME = ?"

    prep_stmt = ibm_db.prepare(conn, sql)

    ibm_db.bind_param(prepare_stmt, 1, session['name'])

    ibm_db.execute(prepare_stmt)    dictionary = ibm_db.fetch_both(prepare_stmt)

    while dictionary != False:

        # print ("The Name is : ", dictionary)

    products.append(dictionary)    dictionary =

    ibm_db.fetch_both(prepare_stmt)    if products:

        return render_template("dashboard/productlist.html", products = products , session = session)

    else:

        return render_template("dashboard/productlist.html")

@app.route('/logout') def logout():

    session.pop('loggedin', None)    session.pop('id', None)

    session.pop('email', None)    session.pop('name', None)

    return redirect(url_for('home'))

@app.route('/list') def list():    users = []

    sql = "SELECT * FROM Users"    stmt =

    ibm_db.exec_immediate(conn, sql)    dictionary =

    ibm_db.fetch_both(stmt)    while dictionary != False:

        # print ("The Name is : ", dictionary)

    users.append(dictionary)    dictionary =

    ibm_db.fetch_both(stmt)

    if users:

        return render_template("list.html", users = users , session = session)    return "No users..."

```


Products.html

```
<!doctype html>

<html class="no-js" lang="zxx">

<head>

    <meta charset="utf-8">

    <meta http-equiv="x-ua-compatible" content="ie=edge">

    <meta name="description" content="">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <!-- Bootstrap Css & Js -->

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet"
integrity="sha384-
1BmE4kWBq78iYhFIdvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3" crossorigin="anonymous">

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-
ka7Sk0GlIn4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBOOLRn5q+8nbTov4+1p"
crossorigin="anonymous"></script>

    <!-- CSS here -->

    <link href="static/css/mystyle.css" rel="Stylesheet" />

<style>

.shadow-demo {    width: 100px;    height:
100px;    background-color: #fff;

}

.shadow-demo-1 {    width: 100px;    height:
100px;    background-color: #ccc;

}

.shadow-demo-2 {    width: 100px;    height:
100px;    background-color: #000;

}
```

```
.mask-custom { background-color: rgba(255, 255, 255, 0.2); border-  
radius: 10px; border: 0px; background-clip: padding-box; box-shadow:  
10px 10px 10px rgba(46, 54, 68, 0.03);  
}  
.custom-1 { backdrop-filter: blur(30px);  
}  
.custom-2 { backdrop-filter: blur(60px);  
}  
.custom-3 { backdrop-filter: blur(40px);  
}  
.custom-4 {  
    backdrop-filter: blur(15px);  
}  
.custom-5 { backdrop-filter: blur(5px);  
}
```

```
.mask-custom-1 { background-color: rgba(0, 0, 0, 0.2); border-radius:  
20px; border: 0px; background-clip: padding-box; box-shadow: 10px  
10px 10px rgba(46, 54, 68, 0.03);  
}  
.custom-6 { backdrop-filter: blur(30px);  
}  
.custom-7 { backdrop-filter: blur(60px);  
}  
.custom-8 { backdrop-filter: blur(40px);  
}  
.custom-9 { backdrop-filter: blur(15px);  
}
```

```

.custom-10 {    backdrop-filter: blur(5px);
}
</style>

<!-- JS here -->

{% block head %} {% endblock %}

<script>    window.watsonAssistantChatOptions = {    integrationID: "633fc278-0dda-417b-9c10-
bd2f300b411a", // The ID of this integration.    region: "jp-tok", // The region your integration is hosted in.
    serviceInstanceID: "b7ec50cd-af28-4bb0-aa53-52dc00c34d4e", // The ID of your service instance.
    onLoad: function(instance) { instance.render(); }
};

    setTimeout(function(){    const
t=document.createElement('script');

        t.src="https://web-chat.global.assistant.watson.appdomain.cloud/versions/" +
(window.watsonAssistantChatOptions.clientVersion || 'latest') +
"/WatsonAssistantChatEntry.js";    document.head.appendChild(t);

    });
</script>
</head>
<body>

<nav class="navbar navbar-expand-lg navbar-light bg-light">
    <div class="container-fluid">
        <a class="navbar-brand" href="/">IMS</a>

        <button class="navbar-toggler" type="button" data-bs-toggle="offcanvas" data-bs-target="#offcanvasNavbar"
aria-controls="offcanvasNavbar">

            <span class="navbar-toggler-icon"></span>

        </button>

        <div class="offcanvas offcanvas-end" tabindex="-1" id="offcanvasNavbar"
arialabelledby="offcanvasNavbarLabel">

            <div class="offcanvas-header">

                <h5 class="offcanvas-title" id="offcanvasNavbarLabel">Offcanvas</h5>

```

```

    <button type="button" class="btn-close text-reset" data-bs-dismiss="offcanvas" arialabel="Close"></button>
</div>
<div class="offcanvas-body">
    <ul class="navbar-nav justify-content-end flex-grow-1 pe-3">
        <li class="nav-item">
            <a class="nav-link" aria-current="page" href="/">Home</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="/register" >Register</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="/login">Login</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="/list" >List</a>
        </li>
    </ul>
</div>
</div>
</nav>
<!--
<nav class="navbar navbar-light bg-light fixed-top">
    <div class="container-fluid">
        Page navigation
        <a class="nav-link" aria-current="page" href="/">Home</a>
        <a class="nav-link" href="/register" >Register</a>
        <a class="nav-link" href="/login">Login</a>
        <a class="nav-link" href="/logout">Logout</a>
        <a class="nav-link" href="/list" >List</a>

```

```
<button class="navbar-toggler" type="button" data-bs-toggle="offcanvas" data-bs-  
target="#offcanvasNavbar" aria-controls="offcanvasNavbar">
```

```
<span class="navbar-toggler-icon"></span>
```

```
</button>
```

```
<div class="offcanvas offcanvas-end" tabindex="-1" id="offcanvasNavbar"  
aria-labelledby="offcanvasNavbarLabel">
```

```
<div class="offcanvas-header">
```

```
<h5 class="offcanvas-title" id="offcanvasNavbarLabel">Offcanvas</h5>
```

```
<button type="button" class="btn-close text-reset" data-bs-dismiss="offcanvas"  
aria-label="Close"></button>
```

```
</div>
```

```
<div class="offcanvas-body">
```

```
<ul class="navbar-nav justify-content-end flex-grow-1 pe-3">
```

```
<li class="nav-item">
```

```
<a class="nav-link" aria-current="page" href="/">Home</a>
```

```
</li>
```

```
<li class="nav-item">
```

```
<a class="nav-link" href="/register" >Register</a>
```

```
</li>
```

```
<li class="nav-item">
```

```
<a class="nav-link" href="/login">Login</a>
```

```
</li>
```

```
<li class="nav-item">
```

```
<a class="nav-link" href="/logout">Logout</a>
```

```
</li>
```

```
<li class="nav-item">
```

```
<a class="nav-link" href="/list" >List</a>
```

```
</li>
```

```
<li class="nav-item dropdown">
```

```
<a class="nav-link dropdown-toggle" href="#" id="offcanvasNavbarDropdown" role="button" data-bs-toggle="dropdown" aria-expanded="false">
```

```
  Dropdown
```

```
</a>
```

```
<ul class="dropdown-menu" aria-labelledby="offcanvasNavbarDropdown">
```

```
<li><a class="dropdown-item" href="#">Action</a></li>
```

```
<li><a class="dropdown-item" href="#">Another action</a></li>
```

```
<li>
```

```
  <hr class="dropdown-divider">
```

```
</li>
```

```
<li><a class="dropdown-item" href="#">Something else here</a></li>
```

```
</ul>
```

```
</li>
```

```
</ul>
```

```
<form class="d-flex">
```

```
  <input class="form-control me-2" type="search" placeholder="Search" arialabel="Search">
```

```
  <button class="btn btn-outline-success" type="submit">Search</button>
```

```
</form>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
</nav> -->
```

```
{% block body %} {% endblock %}
```

```
</body>
```

```
</html>
```

Add product.html

```
<!DOCTYPE html >

<head>

  <meta charset="utf-8">

  <meta http-equiv="x-ua-compatible" content="ie=edge">

  <meta name="description" content="">

  <meta name="viewport" content="width=device-width, initial-scale=1">

  <!-- Bootstrap Css & Js -->

  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet"
integrity="sha384-
1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3" crossorigin="anonymous">

  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-
ka7Sk0GlN4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBOOLRn5q+8nbTov4+1p"
crossorigin="anonymous"></script>

  <style>    html,body
  {        height: 100%;        margin: 0;        font-family: 'Segoe UI', Tahoma,
Geneva, Verdana, sans-serif;
  }
</style>

<!-- CSS here -->

<link href="static/css/mystyle.css" rel="Stylesheet" />

<body>

  <div style="background-image: url('static/img/Secure login-rafiki.png');backgroundposition: center; background-
repeat: no-repeat; background-size: contain; background-repeat:
no-repeat; height: 100%;">

    <h1 class="display-6" style="text-align: center;">We have sent a confirmation mail to your registerd E-mail.</h1>

    <h1 class="display-6" style="text-align: center;"> Please confirm the mail to continue
Registration.</h1>

  </div>

</body>

</html>
```

Config.py

```
import datetime import os from dotenv import load_dotenv load_dotenv()

basedir = os.path.abspath(os.path.dirname(__file__))

APP_SETTINGS = os.getenv('APP_SETTINGS', 'config.DevelopmentConfig') class Config():

    EMAIL_CONFIRMATION_SENDER_EMAIL = os.getenv(
        'EMAIL_CONFIRMATION_SENDER_EMAIL')

    EMAIL_CONFIRMATION_SALT = 'email-confirmation'

    EMAIL_CONFIRMATION_TOKEN_MAX_AGE_SECONDS = 300

    JSON_SORT_KEYS = False

    JWT_ACCESS_TOKEN_EXPIRES = datetime.timedelta(minutes=60)

    SECRET_KEY = os.getenv('SECRET_KEY', os.urandom(32))

    SENDGRID_API_KEY = os.getenv('SENDGRID_API_KEY')

    SQLALCHEMY_TRACK_MODIFICATIONS = False WTF_CSRF_ENABLED = False

class DevelopmentConfig(Config):    DEBUG = True

    JSON_SORT_KEYS = True

    SQLALCHEMY_ECHO = True

    SQLALCHEMY_DATABASE_URI = f'sqlite:///{'os.path.join(basedir, "app.db")}' class ProductionConfig(Config):

    DEBUG = False

    SQLALCHEMY_DATABASE_URI = os.getenv('DB_URL')
```

App.py

```
from turtle import st from flask import Flask, render_template, request, redirect, url_for, session

from markupsafe import escape import ibm_db

conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=54a2f15b-5c0f-46df-
89547e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32733;SECURITY
```



```

=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=lkc93724;PWD=zAzNGa6Da
Nk6xvle",",") import smtplib, ssl ## email.mime subclasses from

email.mime.multipart import MIMEMultipart from email.mime.text

import MIMEText

## The pandas library is only for generating the current date, which is not necessary for sending emails import
pandas as pd from datetime import datetime from flask import Flask app = Flask(__name__)

var_list = [] app.secret_key = 'your secret key'

@app.route('/')

def home(): if not session.get("name"):

    return render_template('home.html') return

render_template('home.html', session = session)

@app.route('/register') def new_student():

    return render_template('Register.html') @app.route('/addrec',methods =

['POST', 'GET']) def addrec():

    if request.method == 'POST': fname = request.form['fname'] lname =

request.form['lname'] cname = request.form['cname'] state =

request.form['state'] city = request.form['city'] mobileno =

request.form['mobileno'] emailid = request.form['emailid'] password =

request.form['password'] pincode = request.form['pincode'] sql = "SELECT *

FROM Users WHERE EMAILID =?"

    stmt = ibm_db.prepare(conn, sql)

    ibm_db.bind_param(stmt,1,emailid)

    ibm_db.execute(stmt) account =

    ibm_db.fetch_assoc(stmt) if account: users = []

        sql = "SELECT * FROM Users" stmt =

        ibm_db.exec_immediate(conn, sql) dictionary =

        ibm_db.fetch_both(stmt) while dictionary != False:

```

```

        # print ("The Name is : ", dictionary)
users.append(dictionary)          dictionary =
ibm_db.fetch_both(stmt)

    return render_template('list.html', msg="You are already a member, please login using your details", users =
users)

    else:

        var_list.append(fname)    var_list.append(lname)
var_list.append(cname)    var_list.append(state)
var_list.append(city)    var_list.append(mobileno)
var_list.append(emailid)    var_list.append(password)
var_list.append(pincodes)

        bodytemp = r"D:\IBM\GUIDED PROJECT\INVENTORY MANAGEMENT SYSTEM FOR RETAILERS\SPRINT
2\templates\email.html"

        with open(bodytemp, "r", encoding='utf-8') as f:

            html= f.read()

        # Set up the email addresses and password. Please replace below with your email address and password
email_from = 'padhu10a@gmail.com'    epassword = 'rbjibzkssszsbrjo'    email_to = emailid

        # Generate today's date to be included in the email Subject    date_str =
pd.Timestamp.today().strftime('%Y-%m-%d')

        # Create a MIMEMultipart class, and set up the From, To, Subject fields    email_message =
MIMEMultipart()    email_message['From'] = email_from

        email_message['To'] = email_to    email_message['Subject'] = f'Report email -
{date_str}'

        # Attach the html doc defined earlier, as a MIMEText html content type to the MIME message
email_message.attach(MIMEText(html, "html"))

        # Convert it as a string    email_string = email_message.as_string()    # Connect to the Gmail SMTP
server and Send Email    context = ssl.create_default_context()    with
smtplib.SMTP_SSL("smtp.gmail.com", 465, context=context) as server:

```

```

        server.login(email_from, epassword)        server.sendmail(email_from,
email_to, email_string)    return render_template('notify.html')

@app.route('/confirm') def confirmation():

    insert_sql = "INSERT INTO Users (FIRSTNAME, LASTNAME, COMPANYNAME,
STATE, CITY, MOBILENO, EMAILID, PASSWORD, PINCODE) VALUES
(?,?,?,?,?,?,?,?)"    prep_stmt = ibm_db.prepare(conn, insert_sql)

    ibm_db.bind_param(prepare_stmt, 1, var_list[0])
    ibm_db.bind_param(prepare_stmt, 2, var_list[1])
    ibm_db.bind_param(prepare_stmt, 3, var_list[2])
    ibm_db.bind_param(prepare_stmt, 4, var_list[3])
    ibm_db.bind_param(prepare_stmt, 5, var_list[4])
    ibm_db.bind_param(prepare_stmt, 6, var_list[5])
    ibm_db.bind_param(prepare_stmt, 7, var_list[6])
    ibm_db.bind_param(prepare_stmt, 8, var_list[7])
    ibm_db.bind_param(prepare_stmt, 9, var_list[8])

    ibm_db.execute(prepare_stmt)    return render_template('confirm.html')

@app.route('/login', methods=['POST', 'GET']) def login():    msg = ""

    if request.method == 'POST' and 'email' in request.form and
'password' in request.form:

        email = request.form['email']        password =
request.form['password']

        sql = "SELECT * FROM Users WHERE EMAILID =? AND PASSWORD =?"

        stmt = ibm_db.prepare(conn, sql)

        ibm_db.bind_param(stmt,1,email)

        ibm_db.bind_param(stmt,2,password)

        ibm_db.execute(stmt)        account = ibm_db.fetch_assoc(stmt)

        if account:

```

```

        session['loggedin'] = True        session['id'] = account['ID']        session['email'] =
account['EMAILID']        session['name'] = account['FIRSTNAME']        msg = 'Logged in
successfully !'        return render_template('dashboard/dashboard.html', msg = msg)

    else:

        msg = 'Incorrect email / password !'    return
render_template('login.html', msg = msg)

@app.route('/dashboard') def dashboard():
    return render_template('dashboard/dashboard.html')

@app.route('/addproduct') def addproduct():
    return render_template('dashboard/addproduct.html')

@app.route('/movement')
def movement():
    products = []

    sql = "SELECT * FROM Products WHERE HOLDERNAME = ?"

    prep_stmt = ibm_db.prepare(conn, sql)

    ibm_db.bind_param(prepare_stmt, 1, session['name'])

    ibm_db.execute(prepare_stmt)    dictionary = ibm_db.fetch_both(prepare_stmt)

    while dictionary != False:

        # print ("The Name is : ", dictionary)

    products.append(dictionary)    dictionary =
ibm_db.fetch_both(prepare_stmt)    if products:

        return render_template("dashboard/movement.html", products = products , session = session)

    else:

        return render_template("dashboard/movement.html")

@app.route('/moveproc', methods = ['POST', 'GET']) def moveproc():    if
request.method == 'POST':

        pname = request.form['pname']        quantityout =
request.form['quantityout']        tow = request.form['to']

```

```

insert_sql = "UPDATE products SET QUANTITYOUT = ?, TO = ? WHERE PRODUCTNAME = ? AND HOLDERNAME = ?;"
prep_stmt = ibm_db.prepare(conn, insert_sql)  ibm_db.bind_param(prepare_stmt,
1,quantityout)  ibm_db.bind_param(prepare_stmt, 2, tow)  ibm_db.bind_param(prepare_stmt, 3,
pname)  ibm_db.bind_param(prepare_stmt, 4, session['name'])  ibm_db.execute(prepare_stmt)

products = []

sql = "SELECT * FROM Products WHERE HOLDERNAME = ?"
prep_stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(prepare_stmt, 1, session['name'])
ibm_db.execute(prepare_stmt)  dictionary = ibm_db.fetch_both(prepare_stmt)
while dictionary != False:
    # print ("The Name is : ", dictionary)
products.append(dictionary)  dictionary =
ibm_db.fetch_both(prepare_stmt)

return render_template('dashboard/movement.html', msg = "Product movement noted!", products =
products) @app.route('/report') def report():
    return render_template('dashboard/report.html')
@app.route('/stockupdate') def stock():
products = []

sql = "SELECT * FROM Products WHERE HOLDERNAME = ?"
prep_stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(prepare_stmt, 1, session['name'])
ibm_db.execute(prepare_stmt)  dictionary = ibm_db.fetch_both(prepare_stmt)
while dictionary != False:
    # print ("The Name is : ", dictionary)
products.append(dictionary)  dictionary =
ibm_db.fetch_both(prepare_stmt)  if products:
    return render_template("dashboard/stockupdate.html", products = products , session = session)
else:

```

```

return render_template("dashboard/stockupdate.html")

@app.route('/proc_delete', methods = ['POST', 'GET']) def proc_delete():

    id = request.args.get('pid')

    delete_sql = "DELETE FROM products WHERE ID = ? AND HOLDERNAME = ?;"

    prep_stmt = ibm_db.prepare(conn, delete_sql)

    ibm_db.bind_param(prepare_stmt, 1, id)      ibm_db.bind_param(prepare_stmt, 2,
session['name'])      ibm_db.execute(prepare_stmt)      products = []

    sql = "SELECT * FROM Products WHERE HOLDERNAME = ?"

    prep_stmt = ibm_db.prepare(conn, sql)

    ibm_db.bind_param(prepare_stmt, 1, session['name'])

    ibm_db.execute(prepare_stmt)      dictionary = ibm_db.fetch_both(prepare_stmt)

    while dictionary != False:

        # print ("The Name is : ", dictionary)

    products.append(dictionary)      dictionary =

    ibm_db.fetch_both(prepare_stmt)

    return render_template('dashboard/stockupdate.html', msg='Product successfully deleted!' , products =
products)

@app.route('/proc_update', methods = ['POST', 'GET']) def proc_update():

    if request.method == 'POST':      pname = request.form['pname']

    quantityin = request.form['quantityin']      pid = request.form['pid']

    update_sql = "UPDATE products SET PRODUCTNAME = ?, QUANTITYIN = ?
WHERE ID = ? AND HOLDERNAME = ?;"

    prep_stmt = ibm_db.prepare(conn, update_sql)

    ibm_db.bind_param(prepare_stmt, 1, pname)

    ibm_db.bind_param(prepare_stmt, 2, quantityin)

    ibm_db.bind_param(prepare_stmt, 3, pid)      ibm_db.bind_param(prepare_stmt,
4, session['name'])      ibm_db.execute(prepare_stmt)      products = []

    sql = "SELECT * FROM Products WHERE HOLDERNAME = ?"

```

```

        prep_stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(prepare_stmt, 1, session['name'])
    ibm_db.execute(prepare_stmt)        dictionary = ibm_db.fetch_both(prepare_stmt)
    while dictionary != False:
        # print ("The Name is : ", dictionary)
    products.append(dictionary)        dictionary =
    ibm_db.fetch_both(prepare_stmt)

    return render_template('dashboard/stockupdate.html', msg='Product successfully updated!' , products =
products)

@app.route('/addproc',methods = ['POST', 'GET']) def addproc():    if request.method ==
'POST':        pname = request.form['pname']        quantity = request.form['quantity']
the_time = datetime.now()        the_time = the_time.replace(second=0, microsecond=0)
sql = "SELECT * FROM Products WHERE HOLDERNAME =?"

        stmt = ibm_db.prepare(conn, sql)        ibm_db.bind_param(stmt,1,session['name'])
    ibm_db.execute(stmt)

    product = ibm_db.fetch_assoc(stmt)
    if product:
        if product['PRODUCTNAME']==pname:
            return render_template('dashboard/addproduct.html', msg="Product already added! Add a new product.")
        else:
            sql ="INSERT INTO Products
(PRODUCTNAME,QUANTITYIN,QUANTITYOUT,TO,DATE,HOLDERNAME)
VALUES (?, ?, ?, ?, ?, ?);"        prep_stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(prepare_stmt, 1, pname)        ibm_db.bind_param(prepare_stmt, 2, quantity)
    ibm_db.bind_param(prepare_stmt, 3, '')        ibm_db.bind_param(prepare_stmt, 4, '')
    ibm_db.bind_param(prepare_stmt, 5, str(the_time))        ibm_db.bind_param(prepare_stmt, 6,
session['name'])        ibm_db.execute(prepare_stmt)        return
    render_template('dashboard/addproduct.html', msg="Product added")

    else:

```

```

        sql="INSERT INTO Products
(PRODUCTNAME,QUANTITYIN,QUANTITYOUT,TO,DATE,HOLDERNAME)
VALUES (?,?,?,?);"      prep_stmt = ibm_db.prepare(conn, sql)

ibm_db.bind_param(prepare_stmt, 1, pname)      ibm_db.bind_param(prepare_stmt, 2, quantity)

ibm_db.bind_param(prepare_stmt, 3, "")      ibm_db.bind_param(prepare_stmt, 4, "")

ibm_db.bind_param(prepare_stmt, 5, str(the_time))      ibm_db.bind_param(prepare_stmt, 6,
session['name'])      ibm_db.execute(prepare_stmt)      return

render_template('dashboard/addproduct.html', msg="Product added") @app.route('/productlist') def
productlist():  products = []

        sql = "SELECT * FROM Products WHERE HOLDERNAME = ?"
        prep_stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(prepare_stmt, 1, session['name'])
        ibm_db.execute(prepare_stmt)  dictionary = ibm_db.fetch_both(prepare_stmt)
        while dictionary != False:
            # print ("The Name is : ", dictionary)
            products.append(dictionary)  dictionary =
            ibm_db.fetch_both(prepare_stmt) if products:
                return render_template("dashboard/productlist.html", products = products , session = session)
            else:
                return render_template("dashboard/productlist.html")

@app.route('/logout') def logout():
    session.pop('loggedin', None)  session.pop('id', None)
    session.pop('email', None)  session.pop('name', None)
    return redirect(url_for('home'))

@app.route('/list') def list():  users = []
    sql = "SELECT * FROM Users"  stmt =
    ibm_db.exec_immediate(conn, sql)  dictionary =
    ibm_db.fetch_both(stmt)  while dictionary != False:

```



```
# print ("The Name is : ", dictionary)
users.append(dictionary)    dictionary =
ibm_db.fetch_both(stmt) if users:
    return render_template("list.html", users = users , session = session) return "No users..."
```

ManageSales.html

```
<html>
<head>
    <meta charset="utf-8">
    <title>MyFlaskApp</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/css/bootstrap.min.css">
</head>
<body>
    {% include 'includes/_navbar.html' %}
    <div class="container mt-4">
        {% include 'includes/_messages.html' %}
        {% block body %}{% endblock %}
    </div>
    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/js/bootstrap.min.js"></script>
    </body>
</html>
```

Addsales.html

```
<html>
<head>
    <meta charset="utf-8">
    <title>MyFlaskApp</title>
```

```

    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/css/bootstrap.min.css">
</head>

<body>

    {% include 'includes/_navbar.html' %}

    <div class="container mt-4">

        {% include 'includes/_messages.html' %}

        {% block body %}{% endblock %}

    </div>

    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/js/bootstrap.min.js"></script>

    </body>
</html>

```

edit_product.html

```

{% extends 'layout.html' %}

{% block body %}

<h1>Edit Product</h1>

{% from "includes/_formhelpers.html" import render_field %}

<form action="" method="POST">

    <div class="form-group">

        {{ render_field(form.product_id, class_="form-control") }}

    </div>

    <div class="form-group">

        {{ render_field(form.product_cost, class_="form-control") }}

    </div>

    <div class="form-group">

        {{ render_field(form.product_num, class_="form-control") }}

    </div>

    <p><input type="submit" value="Update" class="btn btn-primary"></p>

```

```
</form>
```

```
{% endblock %}
```

product_movement. html

```
{% extends 'layout.html' %}
```

```
{% block body %}
```

```
    <h1>Product Movements</h1>
```

```
    <a class="btn btn-success" href="/add_product_movements">Add Product  
Movements</a>
```

```
    <hr>
```

```
    <table class="table table-striped">
```

```
        <thead>
```

```
            <tr>
```

```
                <th>Movement ID</th>
```

```
                <th>Time</th>
```

```
                <th>From Location</th>
```

```
                <th>To Location</th>
```

```
                <th>Product ID</th>
```

```
                <th>Quantity</th>
```

```
            </tr>
```

```
        </thead>
```

```
        <tbody>
```

```
            {% for movement in movements %}
```

```
            <tr>
```

```
                <td>{{movement.MOVEMENT_ID}}</td>
```

```
                <td>{{movement.TIME}}</td>
```

```
                <td>{{movement.FROM_LOCATION}}</td>
```

```
                <td>{{movement.TO_LOCATION}}</td>
```

```

        <td>{{movement.PRODUCT_ID}}</td>        <td>{{movement.QTY}}</td>

        <!--<td><a href="edit_product_movement/{{movement.MOVEMENT_ID}}" class="btn btn-primary pull-
right">Edit</a></td>-->

        <td>

            <form action="{{url_for('delete_product_movements',
id=movement.MOVEMENT_ID)}}" method="POST">

                <input type="hidden" name="method" value="DELETE">

                <input type="submit" value="Delete" class="btn btn-danger">

            </form>

        </td>

    </tr>

    {% endfor %}

</tbody>

</table>

{% endblock %}

```

app.py

```

from flask import Flask, render_template, flash, redirect, url_for, session, request, logging from flask_mysqlldb
import MySQL

from wtforms import Form, StringField, TextAreaField, PasswordField, validators, SelectField, IntegerField
import ibm_db from passlib.hash import sha256_crypt from functools import wraps import win32api

from sendgrid import * #creating an app instance app = Flask(__name__)

conn=ibm_db.connect("DATABASE=bludb;HOSTNAME=;PORT=;SECURITY=SSL;SSL
ServerCertificate=DigiCertGlobalRootCA.crt;UID=;PWD=;", "", "")

#Index @app.route('/') def index():    return

render_template('home.html')

#Products @app.route('/products') def

products():

```

```

    sql = "SELECT * FROM products"    stmt =
ibm_db.prepare(conn, sql)    result=ibm_db.execute(stmt)

products=[]    row = ibm_db.fetch_assoc(stmt)

while(row):

    products.append(row)    row =
ibm_db.fetch_assoc(stmt)    products=tuple(products)

#print(products)    if result>0:

    return render_template('products.html', products = products)

else:

    msg='No products found'    return render_template('products.html',
msg=msg)

#Locations @app.route('/locations') def
locations():

    sql = "SELECT * FROM locations"    stmt =
ibm_db.prepare(conn, sql)    result=ibm_db.execute(stmt)

locations=[]

    row = ibm_db.fetch_assoc(stmt)    while(row):

        locations.append(row)    row =
ibm_db.fetch_assoc(stmt)    locations=tuple(locations)

#print(locations)    if result>0:

    return render_template('locations.html', locations = locations)

else:

    msg='No locations found'    return render_template('locations.html',
msg=msg)

#Product Movements

@app.route('/product_movements') def
product_movements():

```

```

sql = "SELECT * FROM productmovements"  stmt =
ibm_db.prepare(conn, sql)  result=ibm_db.execute(stmt)
movements=[]  row = ibm_db.fetch_assoc(stmt)  while(row):
    movements.append(row)    row =
ibm_db.fetch_assoc(stmt)
movements=tuple(movements)  #print(movements)
if result>0:
    return render_template('product_movements.html', movements = movements)
else:
    msg='No product movements found'    return
render_template('product_movements.html', msg=msg) #Register Form Class class
RegisterForm(Form):
    name = StringField('Name', [validators.Length(min=1, max=50)])  username =
StringField('Username', [validators.Length(min=1, max=25)])  email = StringField('Email',
[validators.length(min=6, max=50)])  password = PasswordField('Password', [
validators.DataRequired(),    validators.EqualTo('confirm', message='Passwords do not match')
])
    confirm = PasswordField('Confirm Password')
#user register
@app.route('/register', methods=['GET','POST']) def register():
    form = RegisterForm(request.form)  if request.method == 'POST' and
form.validate():
        name = form.name.data    email = form.email.data    username = form.username.data
password = sha256_crypt.encrypt(str(form.password.data))  sql1="INSERT INTO users(name, email,
username, password) VALUES(?,?,?,?)"  stmt1 = ibm_db.prepare(conn, sql1)
ibm_db.bind_param(stmt1,1,name)  ibm_db.bind_param(stmt1,2,email)
ibm_db.bind_param(stmt1,3,username)  ibm_db.bind_param(stmt1,4,password)
ibm_db.execute(stmt1)

```

```

        #for flash messages taking parameter and the category of message to be flashed    flash("You are now
registered and can log in", "success")

        #when registration is successful redirect to home

        return redirect(url_for('login'))    return

render_template('register.html', form = form)

#User login

@app.route('/login', methods = ['GET', 'POST']) def login():    if
request.method == 'POST':        #Get form fields        username =
request.form['username']        password_candidate =
request.form['password']        sql1="Select * from users where username =
?"        stmt1 = ibm_db.prepare(conn, sql1)
ibm_db.bind_param(stmt1,1,username)
result=ibm_db.execute(stmt1)        d=ibm_db.fetch_assoc(stmt1)

        if result > 0:            #Get the stored hash

data = d

        password = data['PASSWORD']            #compare passwords            if
sha256_crypt.verify(password_candidate, password):

            #Passed            session['logged_in'] = True
session['username'] = username        flash("you are now logged
in", "success")        return redirect(url_for('dashboard'))

        else:

            error = 'Invalid Login'            return render_template('login.html',
error=error)

        #Close connection

        cur.close()        else:

            error = 'Username not found'            return render_template('login.html',
error=error)    return render_template('login.html')

```

```

#check if user logged in def is_logged_in(f):
@wraps(f) def wrap(*args, **kwargs): if
'logged_in' in session: return f(*args, **kwargs)
else:
flash('Unauthorized, Please login','danger') return
redirect(url_for('login')) return wrap
#Logout
@app.route('/logout') @is_logged_in def logout(): session.clear()
flash("You are now logged out", "success") return
redirect(url_for('login'))
#Dashboard
@app.route('/dashboard') @is_logged_in def
dashboard():
sql2="SELECT product_id, location_id, qty FROM product_balance" sql3="SELECT
location_id FROM locations" stmt2 = ibm_db.prepare(conn, sql2)

stmt3 = ibm_db.prepare(conn, sql3)
result=ibm_db.execute(stmt2) ibm_db.execute(stmt3)
products=[] row = ibm_db.fetch_assoc(stmt2)
while(row):
products.append(row) row =
ibm_db.fetch_assoc(stmt2) products=tuple(products)
locations=[] row2 = ibm_db.fetch_assoc(stmt3)
while(row2):
locations.append(row2) row2 =
ibm_db.fetch_assoc(stmt3) locations=tuple(locations)
locs = [] for i in locations:
locs.append(list(i.values())[0]) if result>0:
return render_template('dashboard.html', products = products, locations = locs)

```



```

else:
    msg='No products found'    return render_template('dashboard.html',
msg=msg)

#Product Form Class class ProductForm(Form):

    product_id = StringField('Product ID', [validators.Length(min=1, max=200)])    product_cost =
StringField('Product Cost', [validators.Length(min=1, max=200)])    product_num = StringField('Product Num',
[validators.Length(min=1, max=200)])

#Add Product

@app.route('/add_product', methods=['GET', 'POST'])

@is_logged_in def add_product():

    form = ProductForm(request.form)    if request.method == 'POST' and
form.validate():

        product_id = form.product_id.data    product_cost = form.product_cost.data    product_num =
form.product_num.data    sql1="INSERT INTO products(product_id, product_cost, product_num) VALUES(?,?,?)"
stmt1 = ibm_db.prepare(conn, sql1)    ibm_db.bind_param(stmt1,1,product_id)
ibm_db.bind_param(stmt1,2,product_cost)    ibm_db.bind_param(stmt1,3,product_num)
ibm_db.execute(stmt1)    flash("Product Added", "success")    return redirect(url_for('products'))    return
render_template('add_product.html', form=form)

#Edit Product

@app.route('/edit_product/<string:id>', methods=['GET', 'POST'])

@is_logged_in def edit_product(id):

    sql1="Select * from products where product_id = ?"    stmt1 =
ibm_db.prepare(conn, sql1)    ibm_db.bind_param(stmt1,1,id)
result=ibm_db.execute(stmt1)    product=ibm_db.fetch_assoc(stmt1)
print(product)    #Get form    form = ProductForm(request.form)

    #populate product form fields    form.product_id.data = product['PRODUCT_ID']
form.product_cost.data = str(product['PRODUCT_COST'])    form.product_num.data
= str(product['PRODUCT_NUM'])    if request.method == 'POST' and form.validate():

```

```

product_id = request.form['product_id']    product_cost =
request.form['product_cost']    product_num = request.form['product_num']

    sql2="UPDATE products SET product_id=?,product_cost=?,product_num=? WHERE product_id=?"    stmt2 =
ibm_db.prepare(conn, sql2)    ibm_db.bind_param(stmt2,1,product_id)
ibm_db.bind_param(stmt2,2,product_cost)    ibm_db.bind_param(stmt2,3,product_num)
ibm_db.bind_param(stmt2,4,id)    ibm_db.execute(stmt2)    flash("Product Updated", "success")    return
redirect(url_for('products'))    return render_template('edit_product.html', form=form)

#Delete Product

@app.route('/delete_product/<string:id>', methods=['POST'])
@is_logged_in def delete_product(id):

    sql2="DELETE FROM products WHERE product_id=?"

    stmt2      =      ibm_db.prepare(conn,      sql2)
ibm_db.bind_param(stmt2,1,id)    ibm_db.execute(stmt2)
flash("Product Deleted", "success")    return
redirect(url_for('products'))

#Location Form Class class
LocationForm(Form):

    location_id = StringField('Location ID', [validators.Length(min=1, max=200)])

#Add Location

@app.route('/add_location', methods=['GET', 'POST'])
@is_logged_in def add_location():

    form = LocationForm(request.form)    if request.method == 'POST' and
form.validate():    location_id = form.location_id.data    sql2="INSERT into
locations VALUES(?)"    stmt2 = ibm_db.prepare(conn, sql2)
ibm_db.bind_param(stmt2,1,location_id)    ibm_db.execute(stmt2)
flash("Location Added", "success")    return redirect(url_for('locations'))

return render_template('add_location.html', form=form)

#Edit Location

@app.route('/edit_location/<string:id>', methods=['GET', 'POST'])

```

```

@is_logged_in def edit_location(id):
    sql2="SELECT * FROM locations where location_id = ?"    stmt2 =
ibm_db.prepare(conn, sql2)    ibm_db.bind_param(stmt2,1,id)
result=ibm_db.execute(stmt2)    location=ibm_db.fetch_assoc(stmt2)

    #Get form    form = LocationForm(request.form)
print(location)

#populate article form fields    form.location_id.data =
location['LOCATION_ID']    if request.method == 'POST' and form.validate():
    location_id = request.form['location_id']    sql2="UPDATE locations SET location_id=?
WHERE location_id=?"    stmt2 = ibm_db.prepare(conn, sql2)
ibm_db.bind_param(stmt2,1,location_id)    ibm_db.bind_param(stmt2,2,id)
ibm_db.execute(stmt2)    flash("Location Updated", "success")    return
redirect(url_for('locations'))    return render_template('edit_location.html', form=form)

#Delete Location

@app.route('/delete_location/<string:id>', methods=['POST'])
@is_logged_in def delete_location(id):
    sql2="DELETE FROM locations WHERE location_id=?"    stmt2 =
ibm_db.prepare(conn, sql2)    ibm_db.bind_param(stmt2,1,id)
ibm_db.execute(stmt2)    flash("Location Deleted", "success")    return
redirect(url_for('locations'))

#Product Movement Form Class class
ProductMovementForm(Form):
    from_location = SelectField('From Location', choices=[])    to_location =
SelectField('To Location', choices=[])    product_id = SelectField('Product ID',
choices=[])    qty = IntegerField('Quantity') class CustomError(Exception):

    pass

#Add Product Movement

```

```

@app.route('/add_product_movements', methods=['GET', 'POST'])
@is_logged_in def add_product_movements():
    form = ProductMovementForm(request.form)    sql2="SELECT
product_id FROM products"    sql3="SELECT location_id FROM
locations"    stmt2 = ibm_db.prepare(conn, sql2)    stmt3 =
ibm_db.prepare(conn, sql3)    result=ibm_db.execute(stmt2)
ibm_db.execute(stmt3)    products=[]    row =
ibm_db.fetch_assoc(stmt2)    while(row):
        products.append(row)    row =
ibm_db.fetch_assoc(stmt2)    products=tuple(products)
locations=[]    row2 = ibm_db.fetch_assoc(stmt3)
while(row2):
        locations.append(row2)    row2 =
ibm_db.fetch_assoc(stmt3)    locations=tuple(locations)
prods = []    for p in products:
        prods.append(list(p.values())[0])    locs = []    for i in
locations:
        locs.append(list(i.values())[0])
form.from_location.choices = [(l,l) for l in locs]
form.from_location.choices.append(("Main
Inventory","Main Inventory"))    form.to_location.choices =
[(l,l) for l in locs]
form.to_location.choices.append(("Main Inventory","Main
Inventory"))    form.product_id.choices = [(p,p) for p in
prods]    if request.method == 'POST' and form.validate():
        from_location = form.from_location.data    to_location =
form.to_location.data    product_id = form.product_id.data
qty = form.qty.data    if from_location==to_location:

```

```

        raise CustomError("Please Give different From and To Locations!!")    elif from_location=="Main
Inventory":    sql2="SELECT * from product_balance where location_id=? and product_id=?"    stmt2
= ibm_db.prepare(conn, sql2)    ibm_db.bind_param(stmt2,1,to_location)
ibm_db.bind_param(stmt2,2,product_id)    result=ibm_db.execute(stmt2)
result=ibm_db.fetch_assoc(stmt2)    print("-----")    print(result)    print("-----")
app.logger.info(result)    if result!=False:    if(len(result))>0:
        Quantity = result["QTY"]    q = Quantity +
qty    sql2="UPDATE product_balance set qty=?
where location_id=? and product_id=?"

        stmt2 = ibm_db.prepare(conn, sql2)    ibm_db.bind_param(stmt2,1,q)
ibm_db.bind_param(stmt2,2,to_location)    ibm_db.bind_param(stmt2,3,product_id)
ibm_db.execute(stmt2)

        sql2="INSERT into productmovements(from_location, to_location, product_id, qty) VALUES(?, ?, ?, ?)"
stmt2 = ibm_db.prepare(conn, sql2)    ibm_db.bind_param(stmt2,1,from_location)
ibm_db.bind_param(stmt2,2,to_location)    ibm_db.bind_param(stmt2,3,product_id)
ibm_db.bind_param(stmt2,4,qty)    ibm_db.execute(stmt2)

    else:

        sql2="INSERT into product_balance(product_id, location_id, qty) values(?, ?, ?)"    stmt2 =
ibm_db.prepare(conn, sql2)    ibm_db.bind_param(stmt2,1,product_id)
ibm_db.bind_param(stmt2,2,to_location)    ibm_db.bind_param(stmt2,3,qty)
ibm_db.execute(stmt2)

        sql2="INSERT into productmovements(from_location, to_location, product_id, qty) VALUES(?, ?, ?, ?)"
stmt2 = ibm_db.prepare(conn, sql2)    ibm_db.bind_param(stmt2,1,from_location)
ibm_db.bind_param(stmt2,2,to_location)    ibm_db.bind_param(stmt2,3,product_id)
ibm_db.bind_param(stmt2,4,qty)    ibm_db.execute(stmt2)

        sql = "select product_num from products where product_id=?"    stmt =
ibm_db.prepare(conn, sql)    ibm_db.bind_param(stmt,1,product_id)
current_num=ibm_db.execute(stmt)    current_num = ibm_db.fetch_assoc(stmt)

```

```

sql2="Update products set product_num=? where product_id=?"      stmt2 =
ibm_db.prepare(conn, sql2)
ibm_db.bind_param(stmt2,1,current_num['PRODUCT_NUM']-qty)
ibm_db.bind_param(stmt2,2,product_id)      ibm_db.execute(stmt2)
alert_num=current_num['PRODUCT_NUM']-qty
    if(alert_num<=0):
        alert("Please update the quantity of the product {}, Atleast {} number of pieces must be added to finish
the pending Product Movements!".format(product_id,-alert_num))      elif to_location=="Main Inventory":
sql2="SELECT * from product_balance where location_id=? and product_id=?"      stmt2 =
ibm_db.prepare(conn, sql2)      ibm_db.bind_param(stmt2,1,from_location)
ibm_db.bind_param(stmt2,2,product_id)      result=ibm_db.execute(stmt2)
result=ibm_db.fetch_assoc(stmt2)  app.logger.info(result)      if result!=False:      if(len(result))>0:
    Quantity = result["QTY"]      q = Quantity -
qty
    sql2="UPDATE product_balance set qty=? where location_id=? and product_id=?"
    stmt2 = ibm_db.prepare(conn, sql2)      ibm_db.bind_param(stmt2,1,q)
ibm_db.bind_param(stmt2,2,to_location)      ibm_db.bind_param(stmt2,3,product_id)
ibm_db.execute(stmt2)
    sql2="INSERT into productmovements(from_location, to_location, product_id, qty) VALUES(?, ?, ?, ?)"
stmt2 = ibm_db.prepare(conn, sql2)      ibm_db.bind_param(stmt2,1,from_location)
ibm_db.bind_param(stmt2,2,to_location)      ibm_db.bind_param(stmt2,3,product_id)
ibm_db.bind_param(stmt2,4,qty)      ibm_db.execute(stmt2)      flash("Product Movement Added",
"success")      sql = "select product_num from products where product_id=?"      stmt =
ibm_db.prepare(conn, sql)      ibm_db.bind_param(stmt,1,product_id)
current_num=ibm_db.execute(stmt)      current_num = ibm_db.fetch_assoc(stmt)
sql2="Update products set product_num=? where product_id=?"      stmt2 = ibm_db.prepare(conn, sql2)
ibm_db.bind_param(stmt2,1,current_num['PRODUCT_NUM']+qty)

```

```

ibm_db.bind_param(stmt2,2,product_id)          ibm_db.execute(stmt2)          alert_num=q
if(alert_num<=0):
    alert("Please Add {} number of {} to {} warehouse!".format(q,product_id,from_location))
else:
    raise CustomError("There is no product named {} in
{}.format(product_id,from_location))          else: #will be executed if both from_location and
to_location are specified          f=0

    sql = "SELECT * from product_balance where location_id=? and product_id=?"          stmt =
ibm_db.prepare(conn, sql)          ibm_db.bind_param(stmt,1,from_location)
ibm_db.bind_param(stmt,2,product_id)          result=ibm_db.execute(stmt)          result =
ibm_db.fetch_assoc(stmt) if result!=False:
    if(len(result))>0:
        Quantity = result["QTY"]          q = Quantity -
qty
        sql2="UPDATE product_balance set qty=? where location_id=? and product_id=?"
        stmt2 = ibm_db.prepare(conn, sql2)          ibm_db.bind_param(stmt2,1,q)
ibm_db.bind_param(stmt2,2,from_location)          ibm_db.bind_param(stmt2,3,product_id)
ibm_db.execute(stmt2)
        f=1          alert_num=q
if(alert_num<=0):
    alert("Please Add {} number of {} to {} warehouse!".format(q,product_id,from_location))
else:
    raise CustomError("There is no product named {} in
{}.format(product_id,from_location))
    if(f==1):
        sql = "SELECT * from product_balance where location_id=? and product_id=?"          stmt =
ibm_db.prepare(conn, sql)          ibm_db.bind_param(stmt,1,to_location)

```

```

        ibm_db.bind_param(stmt,2,product_id)
result=ibm_db.execute(stmt)          result =
ibm_db.fetch_assoc(stmt)          if result!=False:
if(len(result))>0:
        Quantity = result["QTY"]          q = Quantity
+ qty

        sql2="UPDATE product_balance set qty=? where location_id=? and product_id=?"
        stmt2 = ibm_db.prepare(conn, sql2)          ibm_db.bind_param(stmt2,1,q)
ibm_db.bind_param(stmt2,2,to_location)          ibm_db.bind_param(stmt2,3,product_id)
ibm_db.execute(stmt2)

        else:
                sql2="INSERT into product_balance(product_id, location_id, qty) values(?, ?, ?)"
stmt2 = ibm_db.prepare(conn, sql2)          ibm_db.bind_param(stmt2,1,product_id)
ibm_db.bind_param(stmt2,2,to_location)          ibm_db.bind_param(stmt2,3,qty)
ibm_db.execute(stmt2)

        sql2="INSERT into productmovements(from_location, to_location, product_id, qty) VALUES(?, ?, ?, ?)"
stmt2 = ibm_db.prepare(conn, sql2)          ibm_db.bind_param(stmt2,1,from_location)
ibm_db.bind_param(stmt2,2,to_location)          ibm_db.bind_param(stmt2,3,product_id)
ibm_db.bind_param(stmt2,4,qty)          ibm_db.execute(stmt2)

        flash("Product Movement Added", "success")
render_template('products.html',form=form)    return
redirect(url_for('product_movements'))    return
render_template('add_product_movements.html', form=form)

#Delete Product Movements
@app.route('/delete_product_movements/<string:id>', methods=['POST'])
@is_logged_in def delete_product_movements(id):
        sql2="DELETE FROM productmovements WHERE movement_id=?"

```



```

    stmt2 = ibm_db.prepare(conn, sql2)
ibm_db.bind_param(stmt2,1,id)    ibm_db.execute(stmt2)

flash("Product Movement Deleted", "success")    return

redirect(url_for('product_movements')) if __name__ == '__main__':

app.secret_key = "secret123"

    #when the debug mode is on, we do not need to restart the server again and again    app.run(debug=True)

```

config.py

```

from flask import Flask, render_template, flash, redirect, url_for, session, request, logging from flask_mysqlldb
import MySQL

from wtforms import Form, StringField, TextAreaField, PasswordField, validators, SelectField, IntegerField

import ibm_db from passlib.hash import sha256_crypt from functools import wraps import win32api

from sendgrid import * #creating an app instance

app = Flask(__name__)

conn=ibm_db.connect("DATABASE=bludb;HOSTNAME=;PORT=;SECURITY=SSL;SSL
ServerCertificate=DigiCertGlobalRootCA.crt;UID=;PWD=;",",",")

#Index @app.route('/') def index():    return

render_template('home.html')

#Products

@app.route('/products') def products():

    sql = "SELECT * FROM products"    stmt =

ibm_db.prepare(conn, sql)    result=ibm_db.execute(stmt)

products=[]    row = ibm_db.fetch_assoc(stmt)

while(row):

    products.append(row)    row =

ibm_db.fetch_assoc(stmt)    products=tuple(products)

#print(products)    if result>0:

```

```

        return render_template('products.html', products = products)
    else:
        msg='No products found'    return render_template('products.html',
msg=msg)
#Locations
@app.route('/locations')
def locations():
    sql = "SELECT * FROM locations"    stmt =
ibm_db.prepare(conn, sql)    result=ibm_db.execute(stmt)
locations=[]    row = ibm_db.fetch_assoc(stmt)
while(row):
    locations.append(row)    row =
ibm_db.fetch_assoc(stmt)    locations=tuple(locations)
#print(locations)    if result>0:
    return render_template('locations.html', locations = locations)
else:
    msg='No locations found'    return render_template('locations.html',
msg=msg)
#Product Movements @app.route('/product_movements')
def product_movements():
    sql = "SELECT * FROM productmovements"    stmt =
ibm_db.prepare(conn, sql)    result=ibm_db.execute(stmt)
movements=[]    row = ibm_db.fetch_assoc(stmt)    while(row):
    movements.append(row)    row =
ibm_db.fetch_assoc(stmt)
movements=tuple(movements)
#print(movements)
if result>0:

```

```

    return render_template('product_movements.html', movements = movements)
else:
    msg='No product movements found'    return
render_template('product_movements.html', msg=msg)
#Register Form Class class RegisterForm(Form):
    name = StringField('Name', [validators.Length(min=1, max=50)])    username =
StringField('Username', [validators.Length(min=1, max=25)])    email = StringField('Email',
[validators.length(min=6, max=50)])    password = PasswordField('Password', [
validators.DataRequired(),    validators.EqualTo('confirm', message='Passwords do not match')
])
    confirm = PasswordField('Confirm Password')
#user register
@app.route('/register', methods=['GET','POST']) def register():
    form = RegisterForm(request.form)    if request.method == 'POST' and
form.validate():
        name = form.name.data    email = form.email.data    username = form.username.data
password = sha256_crypt.encrypt(str(form.password.data))    sql1="INSERT INTO users(name, email,
username, password) VALUES(?,?,?,?)"    stmt1 = ibm_db.prepare(conn, sql1)
ibm_db.bind_param(stmt1,1,name)    ibm_db.bind_param(stmt1,2,email)
ibm_db.bind_param(stmt1,3,username)

    ibm_db.bind_param(stmt1,4,password)    ibm_db.execute(stmt1)
    #for flash messages taking parameter and the category of message to be flashed    flash("You are
now registered and can log in", "success")    #when registration is successful redirect to home
return redirect(url_for('login'))    return render_template('register.html', form = form)
#User login
@app.route('/login', methods = ['GET', 'POST']) def login():
    if request.method == 'POST':    #Get form fields    username =
request.form['username']    password_candidate =

```

```

request.form['password']    sql1="Select * from users where username
= ?"    stmt1 = ibm_db.prepare(conn, sql1)
ibm_db.bind_param(stmt1,1,username)
result=ibm_db.execute(stmt1)    d=ibm_db.fetch_assoc(stmt1)

    if result > 0:

        #Get the stored hash        data = d

        password = data['PASSWORD']

        #compare passwords        if sha256_crypt.verify(password_candidate,
password):

            #Passed        session['logged_in'] = True
session['username'] = username        flash("you are now logged
in","success")        return redirect(url_for('dashboard'))

    else:

        error = 'Invalid Login'        return render_template('login.html',
error=error)

        #Close connection        cur.close()
else:

    error = 'Username not found'        return
render_template('login.html', error=error)    return
render_template('login.html') #check if user logged in def is_logged_in(f):
@wraps(f)    def wrap(*args, **kwargs):        if 'logged_in' in session:
return f(*args, **kwargs)

    else:

        flash('Unauthorized, Please login','danger')        return
redirect(url_for('login'))    return wrap
#Logout

```

```

@app.route('/logout') @is_logged_in def logout(): session.clear()
flash("You are now logged out", "success") return
redirect(url_for('login'))

#Dashboard

@app.route('/dashboard')
@is_logged_in def dashboard():
    sql2="SELECT product_id, location_id, qty FROM product_balance" sql3="SELECT
location_id FROM locations" stmt2 = ibm_db.prepare(conn, sql2) stmt3 =
ibm_db.prepare(conn, sql3) result=ibm_db.execute(stmt2) ibm_db.execute(stmt3)
products=[] row = ibm_db.fetch_assoc(stmt2) while(row):
    products.append(row) row =
ibm_db.fetch_assoc(stmt2) products=tuple(products)
locations=[] row2 = ibm_db.fetch_assoc(stmt3)
while(row2):
    locations.append(row2) row2 =
ibm_db.fetch_assoc(stmt3) locations=tuple(locations)
locs = [] for i in locations:
    locs.append(list(i.values())[0]) if result>0:
        return render_template('dashboard.html', products = products, locations = locs)
    else:
        msg='No products found' return render_template('dashboard.html',
msg=msg)

#Product Form Class
class ProductForm(Form):
    product_id = StringField('Product ID', [validators.Length(min=1, max=200)]) product_cost =
StringField('Product Cost', [validators.Length(min=1, max=200)]) product_num = StringField('Product Num',
[validators.Length(min=1, max=200)])

#Add Product

```

```

@app.route('/add_product', methods=['GET', 'POST'])
@is_logged_in def add_product():
    form = ProductForm(request.form)    if request.method == 'POST' and
form.validate():
    product_id = form.product_id.data    product_cost =
form.product_cost.data    product_num = form.product_num.data
    sql1="INSERT INTO products(product_id, product_cost, product_num) VALUES(?,?,?)"
    stmt1 = ibm_db.prepare(conn, sql1)
ibm_db.bind_param(stmt1,1,product_id)
ibm_db.bind_param(stmt1,2,product_cost)
ibm_db.bind_param(stmt1,3,product_num)    ibm_db.execute(stmt1)
flash("Product Added", "success")    return redirect(url_for('products'))
return render_template('add_product.html', form=form)

#Edit Product

@app.route('/edit_product/<string:id>', methods=['GET', 'POST'])
@is_logged_in def edit_product(id):
    sql1="Select * from products where product_id = ?"    stmt1 =
ibm_db.prepare(conn, sql1)    ibm_db.bind_param(stmt1,1,id)
result=ibm_db.execute(stmt1)    product=ibm_db.fetch_assoc(stmt1)
    print(product)    #Get form    form = ProductForm(request.form) #populate
product form fields    form.product_id.data = product['PRODUCT_ID']
form.product_cost.data = str(product['PRODUCT_COST'])    form.product_num.data
= str(product['PRODUCT_NUM'])    if request.method == 'POST' and form.validate():
product_id = request.form['product_id']    product_cost =
request.form['product_cost']    product_num = request.form['product_num']
    sql2="UPDATE products SET product_id=?,product_cost=?,product_num=? WHERE product_id=?"    stmt2 =
ibm_db.prepare(conn, sql2)    ibm_db.bind_param(stmt2,1,product_id)
ibm_db.bind_param(stmt2,2,product_cost)    ibm_db.bind_param(stmt2,3,product_num)

```

```

ibm_db.bind_param(stmt2,4,id)    ibm_db.execute(stmt2)    flash("Product Updated", "success")    return
redirect(url_for('products'))    return render_template('edit_product.html', form=form)

#Delete Product

@app.route('/delete_product/<string:id>', methods=['POST'])

@is_logged_in def delete_product(id):

sql2="DELETE FROM products WHERE

product_id=?"

    stmt2    =    ibm_db.prepare(conn,    sql2)
ibm_db.bind_param(stmt2,1,id)    ibm_db.execute(stmt2)
flash("Product Deleted", "success")    return
redirect(url_for('products'))

#Location Form Class class LocationForm(Form):

    location_id = StringField('Location ID', [validators.Length(min=1, max=200)])

#Add Location

@app.route('/add_location', methods=['GET', 'POST'])

@is_logged_in def add_location():

    form = LocationForm(request.form)    if request.method == 'POST' and
form.validate():    location_id = form.location_id.data    sql2="INSERT into
locations VALUES(?)"    stmt2 = ibm_db.prepare(conn, sql2)
ibm_db.bind_param(stmt2,1,location_id)    ibm_db.execute(stmt2)
flash("Location Added", "success")    return redirect(url_for('locations'))
return render_template('add_location.html', form=form)

#Edit Location

@app.route('/edit_location/<string:id>', methods=['GET', 'POST'])

@is_logged_in def edit_location(id):

    sql2="SELECT * FROM locations where location_id = ?"    stmt2 =
ibm_db.prepare(conn, sql2)    ibm_db.bind_param(stmt2,1,id)
result=ibm_db.execute(stmt2)    location=ibm_db.fetch_assoc(stmt2)

```

```

    #Get form    form = LocationForm(request.form)

print(location)

    #populate article form fields    form.location_id.data = location['LOCATION_ID'] if
request.method == 'POST' and form.validate():    location_id = request.form['location_id']

sql2="UPDATE locations SET location_id=? WHERE location_id=?"    stmt2 =

ibm_db.prepare(conn, sql2)    ibm_db.bind_param(stmt2,1,location_id)

ibm_db.bind_param(stmt2,2,id)    ibm_db.execute(stmt2)    flash("Location Updated",
"success")    return redirect(url_for('locations'))    return render_template('edit_location.html',
form=form)

#Delete Location

@app.route('/delete_location/<string:id>', methods=['POST'])

@is_logged_in def delete_location(id):

    sql2="DELETE FROM locations WHERE location_id=?"    stmt2 =

ibm_db.prepare(conn, sql2)    ibm_db.bind_param(stmt2,1,id)

ibm_db.execute(stmt2)    flash("Location Deleted", "success")    return
redirect(url_for('locations'))

#Product Movement Form Class

class ProductMovementForm(Form):

    from_location = SelectField('From Location', choices=[])    to_location =
SelectField('To Location', choices=[])    product_id = SelectField('Product ID',
choices=[])    qty = IntegerField('Quantity') class CustomError(Exception):

    pass

#Add Product Movement

@app.route('/add_product_movements', methods=['GET', 'POST'])

@is_logged_in def add_product_movements():

    form = ProductMovementForm(request.form)    sql2="SELECT
product_id FROM products"    sql3="SELECT location_id FROM
locations"    stmt2 = ibm_db.prepare(conn, sql2)    stmt3 =

```



```

ibm_db.prepare(conn, sql3)  result=ibm_db.execute(stmt2)
ibm_db.execute(stmt3)  products=[]  row =
ibm_db.fetch_assoc(stmt2)  while(row):
    products.append(row)  row =
ibm_db.fetch_assoc(stmt2)  products=tuple(products)
locations=[]  row2 = ibm_db.fetch_assoc(stmt3)
while(row2):
    locations.append(row2)  row2 =
ibm_db.fetch_assoc(stmt3)  locations=tuple(locations)
prods = []  for p in products:
    prods.append(list(p.values())[0])  locs = []  for i in
locations:
    locs.append(list(i.values())[0]) form.from_location.choices = [(l,l) for l in locs]
form.from_location.choices.append(("Main Inventory","Main Inventory"))
form.to_location.choices = [(l,l) for l in locs]  form.to_location.choices.append(("Main
Inventory","Main Inventory"))  form.product_id.choices = [(p,p) for p in prods]  if
request.method == 'POST' and form.validate():
    from_location = form.from_location.data  to_location =
form.to_location.data  product_id = form.product_id.data
qty = form.qty.data  if from_location==to_location:
    raise CustomError("Please Give different From and To Locations!!")

    elif from_location=="Main Inventory":  sql2="SELECT * from product_balance where location_id=?
and product_id=?"  stmt2 = ibm_db.prepare(conn, sql2)
ibm_db.bind_param(stmt2,1,to_location)  ibm_db.bind_param(stmt2,2,product_id)
result=ibm_db.execute(stmt2)  result=ibm_db.fetch_assoc(stmt2)  print("-----")
print(result)

```

```

        print("-----")
app.logger.info(result)      if result!=False:
if(len(result))>0:
        Quantity = result["QTY"]      q = Quantity +
qty
        sql2="UPDATE product_balance set qty=? where location_id=? and product_id=?"
        stmt2 = ibm_db.prepare(conn, sql2)      ibm_db.bind_param(stmt2,1,q)
ibm_db.bind_param(stmt2,2,to_location)      ibm_db.bind_param(stmt2,3,product_id)
ibm_db.execute(stmt2)
        sql2="INSERT into productmovements(from_location, to_location, product_id, qty) VALUES(?, ?, ?, ?)"
stmt2 = ibm_db.prepare(conn, sql2)      ibm_db.bind_param(stmt2,1,from_location)
ibm_db.bind_param(stmt2,2,to_location)      ibm_db.bind_param(stmt2,3,product_id)
ibm_db.bind_param(stmt2,4,qty)      ibm_db.execute(stmt2)
else:
        sql2="INSERT into product_balance(product_id, location_id, qty) values(?, ?, ?)"      stmt2 =
ibm_db.prepare(conn, sql2)      ibm_db.bind_param(stmt2,1,product_id)
ibm_db.bind_param(stmt2,2,to_location)      ibm_db.bind_param(stmt2,3,qty)
ibm_db.execute(stmt2)
        sql2="INSERT into productmovements(from_location, to_location, product_id, qty) VALUES(?, ?, ?, ?)"
        stmt2 = ibm_db.prepare(conn, sql2)
ibm_db.bind_param(stmt2,1,from_location)      ibm_db.bind_param(stmt2,2,to_location)
ibm_db.bind_param(stmt2,3,product_id)      ibm_db.bind_param(stmt2,4,qty)
ibm_db.execute(stmt2)      sql = "select product_num from products where product_id=?"
stmt = ibm_db.prepare(conn, sql)      ibm_db.bind_param(stmt,1,product_id)
current_num=ibm_db.execute(stmt)      current_num = ibm_db.fetch_assoc(stmt)
sql2="Update products set product_num=? where product_id=?"      stmt2 =
ibm_db.prepare(conn, sql2)
ibm_db.bind_param(stmt2,1,current_num['PRODUCT_NUM']-qty)

```

```

ibm_db.bind_param(stmt2,2,product_id)      ibm_db.execute(stmt2)
alert_num=current_num['PRODUCT_NUM']-qty

if(alert_num<=0):
    alert("Please update the quantity of the product {}, Atleast {} number of pieces must be added to finish
the pending Product Movements!".format(product_id,-alert_num))      elif to_location=="Main Inventory":
sql2="SELECT * from product_balance where location_id=? and product_id=?"      stmt2 =
ibm_db.prepare(conn, sql2)      ibm_db.bind_param(stmt2,1,from_location)
ibm_db.bind_param(stmt2,2,product_id)      result=ibm_db.execute(stmt2)
result=ibm_db.fetch_assoc(stmt2)  app.logger.info(result)      if result!=False:

    if(len(result))>0:
        Quantity = result["QTY"]      q = Quantity -
qty
        sql2="UPDATE product_balance set qty=? where location_id=? and product_id=?"
        stmt2 = ibm_db.prepare(conn, sql2)      ibm_db.bind_param(stmt2,1,q)
ibm_db.bind_param(stmt2,2,to_location)      ibm_db.bind_param(stmt2,3,product_id)
ibm_db.execute(stmt2)

        sql2="INSERT into productmovements(from_location, to_location, product_id, qty) VALUES(?, ?, ?, ?)"
stmt2 = ibm_db.prepare(conn, sql2)      ibm_db.bind_param(stmt2,1,from_location)
ibm_db.bind_param(stmt2,2,to_location)      ibm_db.bind_param(stmt2,3,product_id)
ibm_db.bind_param(stmt2,4,qty)      ibm_db.execute(stmt2)      flash("Product Movement Added",
"success")      sql = "select product_num from products where product_id=?"      stmt =
ibm_db.prepare(conn, sql)      ibm_db.bind_param(stmt,1,product_id)
current_num=ibm_db.execute(stmt)      current_num = ibm_db.fetch_assoc(stmt)
sql2="Update products set product_num=? where product_id=?"      stmt2 = ibm_db.prepare(conn, sql2)
ibm_db.bind_param(stmt2,1,current_num['PRODUCT_NUM']+qty)
ibm_db.bind_param(stmt2,2,product_id)      ibm_db.execute(stmt2)      alert_num=q

if(alert_num<=0):
    alert("Please Add {} number of {} to {} warehouse!".format(q,product_id,from_location))

```

```

else:

    raise CustomError("There is no product named {} in
{}.format(product_id,from_location))           else: #will be executed if both from_location and
to_location are specified

    f=0

    sql = "SELECT * from product_balance where location_id=? and product_id=?"           stmt =
ibm_db.prepare(conn, sql)           ibm_db.bind_param(stmt,1,from_location)
ibm_db.bind_param(stmt,2,product_id)           result=ibm_db.execute(stmt)           result =
ibm_db.fetch_assoc(stmt) if result!=False:

    if(len(result))>0:

        Quantity = result["QTY"]           q = Quantity -
qty

        sql2="UPDATE product_balance set qty=? where location_id=? and product_id=?"
        stmt2 = ibm_db.prepare(conn, sql2)           ibm_db.bind_param(stmt2,1,q)
ibm_db.bind_param(stmt2,2,from_location)           ibm_db.bind_param(stmt2,3,product_id)
ibm_db.execute(stmt2)

        f=1           alert_num=q

if(alert_num<=0):

    alert("Please Add {} number of {} to {} warehouse!".format(q,product_id,from_location))

else:

    raise CustomError("There is no product named {} in
{}.format(product_id,from_location))

    if(f==1):

        sql = "SELECT * from product_balance where location_id=? and product_id=?"           stmt =
ibm_db.prepare(conn, sql)           ibm_db.bind_param(stmt,1,to_location)
ibm_db.bind_param(stmt,2,product_id)           result=ibm_db.execute(stmt)           result =
ibm_db.fetch_assoc(stmt)           if result!=False:           if(len(result))>0:

            Quantity = result["QTY"]           q = Quantity
+ qty

```

```

        sql2="UPDATE product_balance set qty=? where location_id=? and product_id=?"
        stmt2 = ibm_db.prepare(conn, sql2)                ibm_db.bind_param(stmt2,1,q)
ibm_db.bind_param(stmt2,2,to_location)                ibm_db.bind_param(stmt2,3,product_id)
ibm_db.execute(stmt2)

    else:

        sql2="INSERT into product_balance(product_id, location_id, qty) values(?, ?, ?)"
        stmt2 = ibm_db.prepare(conn, sql2)                ibm_db.bind_param(stmt2,1,product_id)
ibm_db.bind_param(stmt2,2,to_location)                ibm_db.bind_param(stmt2,3,qty)
ibm_db.execute(stmt2)

        sql2="INSERT into productmovements(from_location, to_location, product_id, qty) VALUES(?, ?, ?, ?)"
        stmt2 = ibm_db.prepare(conn, sql2)                ibm_db.bind_param(stmt2,1,from_location)
ibm_db.bind_param(stmt2,2,to_location)                ibm_db.bind_param(stmt2,3,product_id)
ibm_db.bind_param(stmt2,4,qty)                ibm_db.execute(stmt2)    flash("Product Movement Added",
"success")    render_template('products.html',form=form)    return redirect(url_for('product_movements'))
return render_template('add_product_movements.html', form=form)

#Delete Product Movements

@app.route('/delete_product_movements/<string:id>', methods=['POST'])
@is_logged_in def delete_product_movements(id):

    sql2="DELETE FROM productmovements WHERE movement_id=?"

    stmt2 = ibm_db.prepare(conn, sql2)    ibm_db.bind_param(stmt2,1,id)
ibm_db.execute(stmt2)


    flash("Product Movement Deleted", "success")    return
redirect(url_for('product_movements')) if __name__ == '__main__':

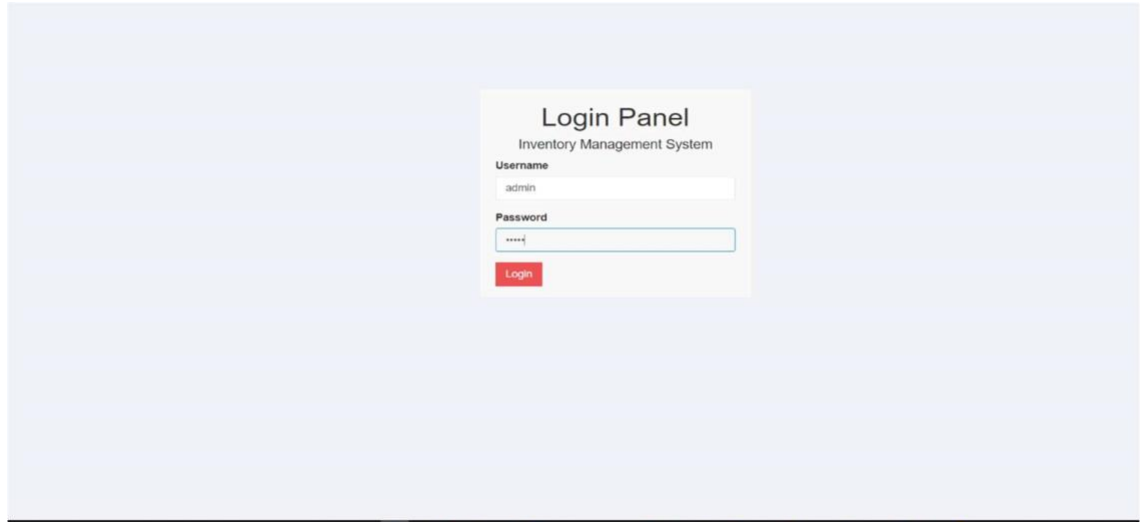
app.secret_key = "secret123"

#when the debug mode is on, we do not need to restart the server again and again    app.run(debug=True)

```

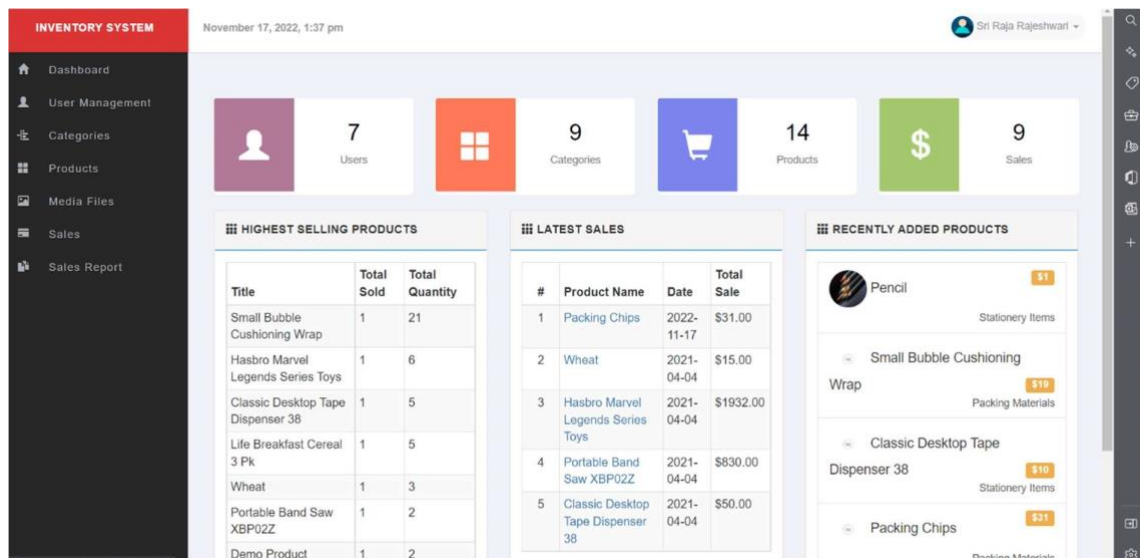
7.4 Outputs

As an admin, I can login and manage the inventory

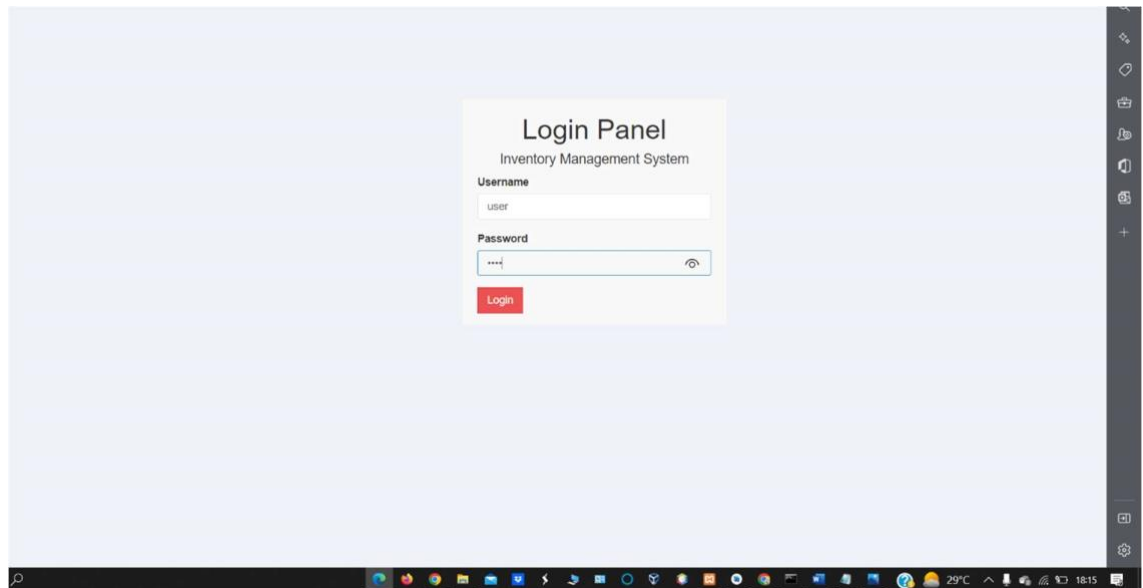


DASHBOARD

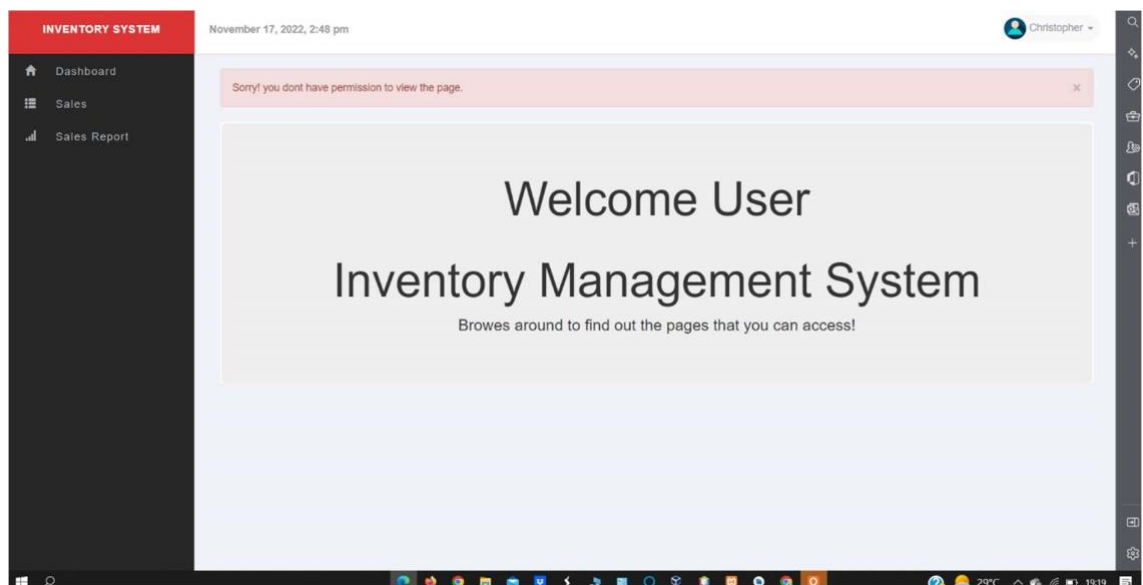
I have successfully login to the next page



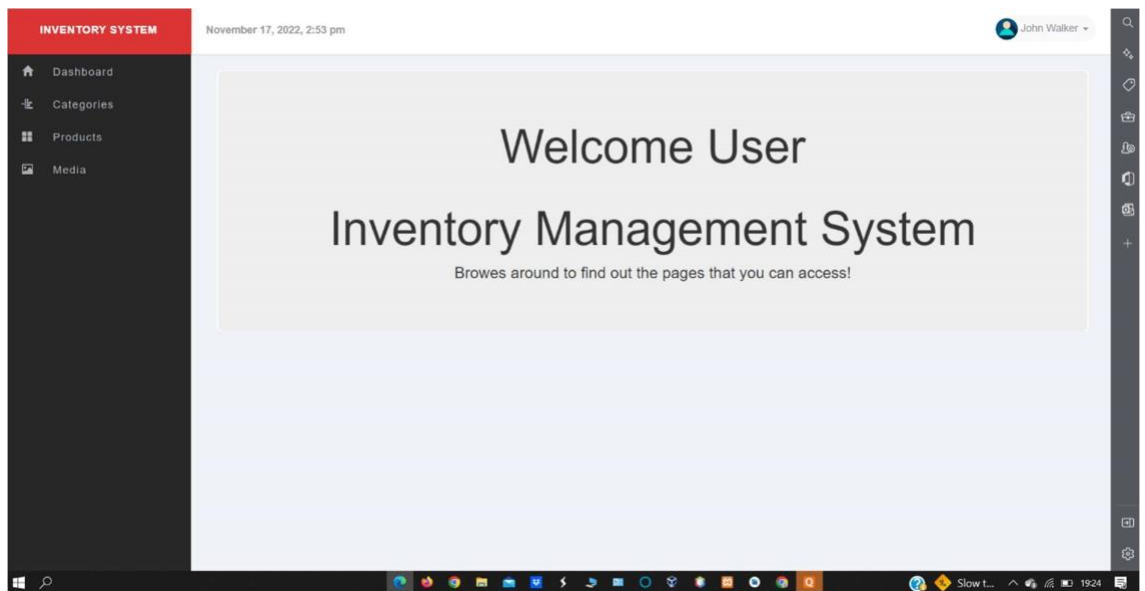
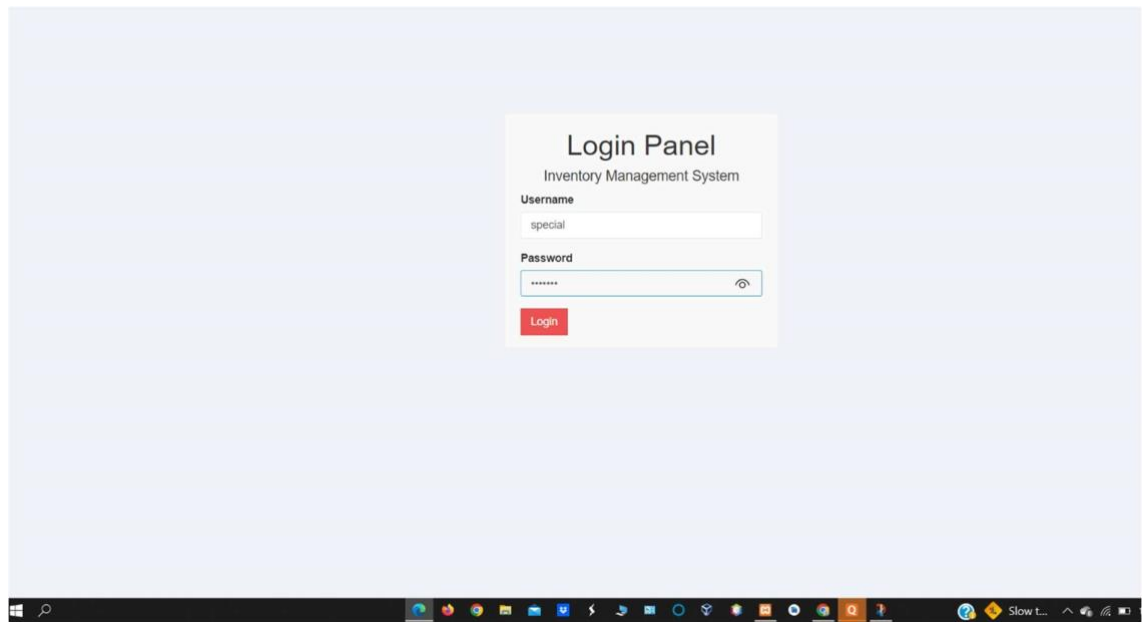
We can login for user account also



User have their respective roles



Special user have only access to products and media



As an admin, I can manage and add users

The screenshot shows the 'INVENTORY SYSTEM' interface. The left sidebar contains navigation links: Dashboard, User Management, Categories, Products, Media Files, Sales, and Sales Report. The main content area is titled 'USERS' and includes an 'ADD NEW USER' button. Below the button is a table listing existing users.

#	Name	Username	User Role	Status	Last Login	Actions
1	Christopher	User	User	Active	November 17, 2022, 2:59:36 pm	[Edit] [Delete]
2	John Walker	Special	Special	Active	November 17, 2022, 3:01:50 pm	[Edit] [Delete]
3	Kevin	Kevin	User	Active	April 4, 2021, 7:54:29 pm	[Edit] [Delete]
4	Natie Williams	Natie	User	Active		[Edit] [Delete]
5	Sri Raja Rajeshwari	Admin	Admin	Active	November 17, 2022, 3:05:11 pm	[Edit] [Delete]

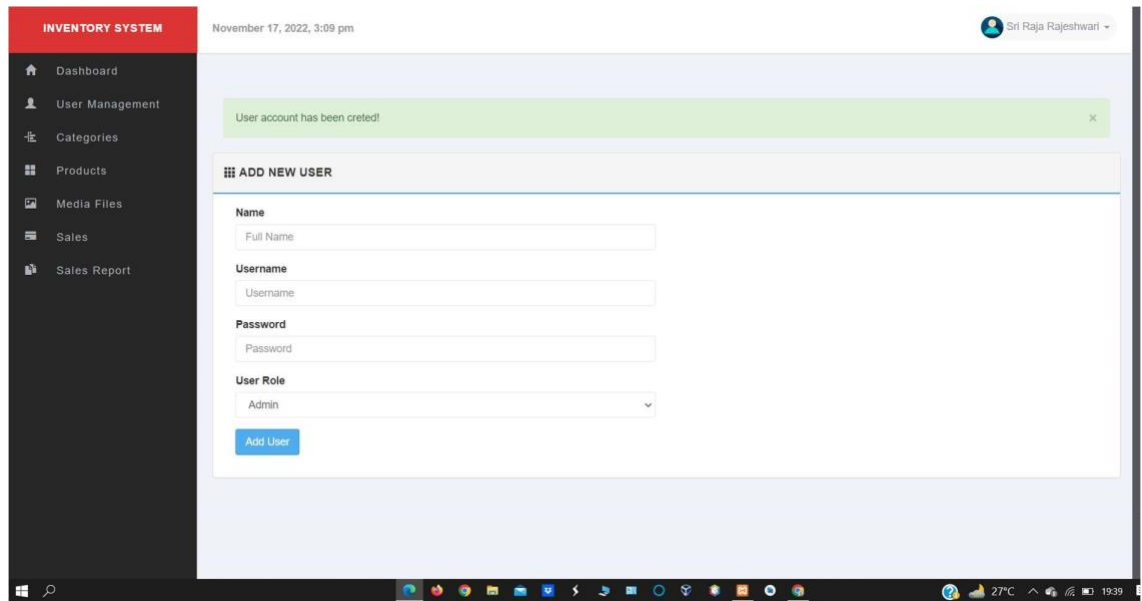
Adding new users by giving correct credentials

The screenshot shows the 'INVENTORY SYSTEM' interface with the 'ADD NEW USER' form open. The form contains the following fields:

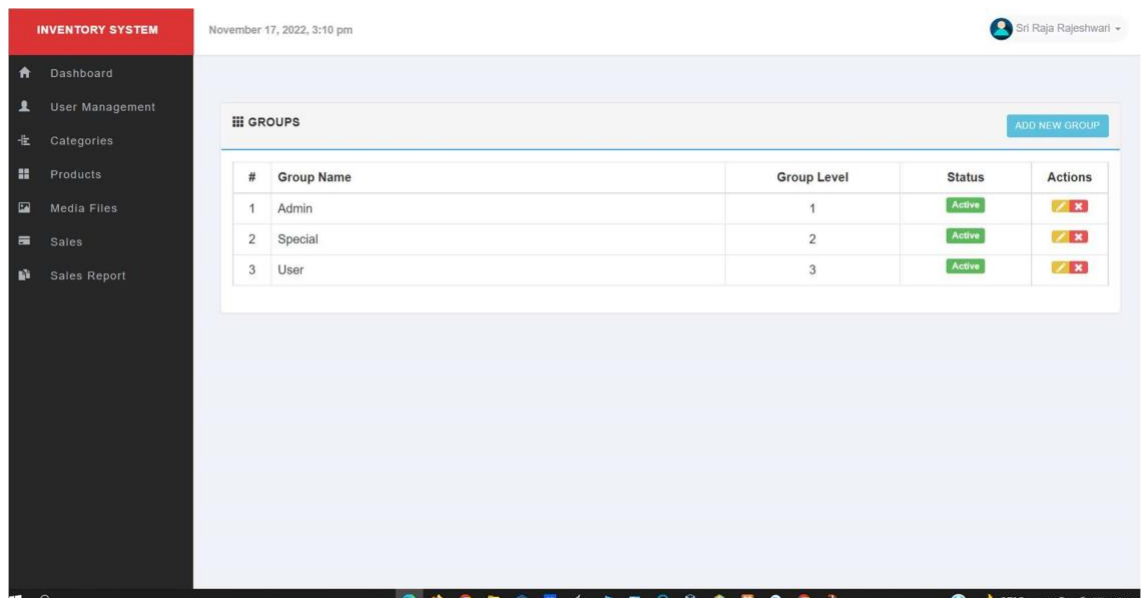
- Name:** Muglia
- Username:** user
- Password:** (masked with dots)
- User Role:** User (selected from a dropdown menu)

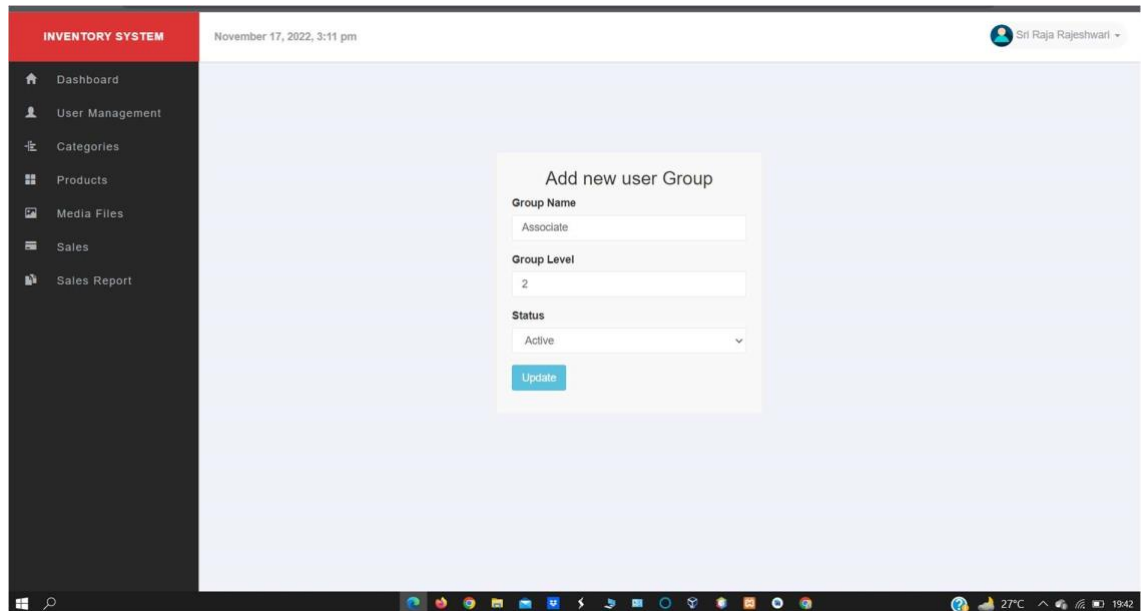
An 'Add User' button is located at the bottom of the form.

User account has been created successfully

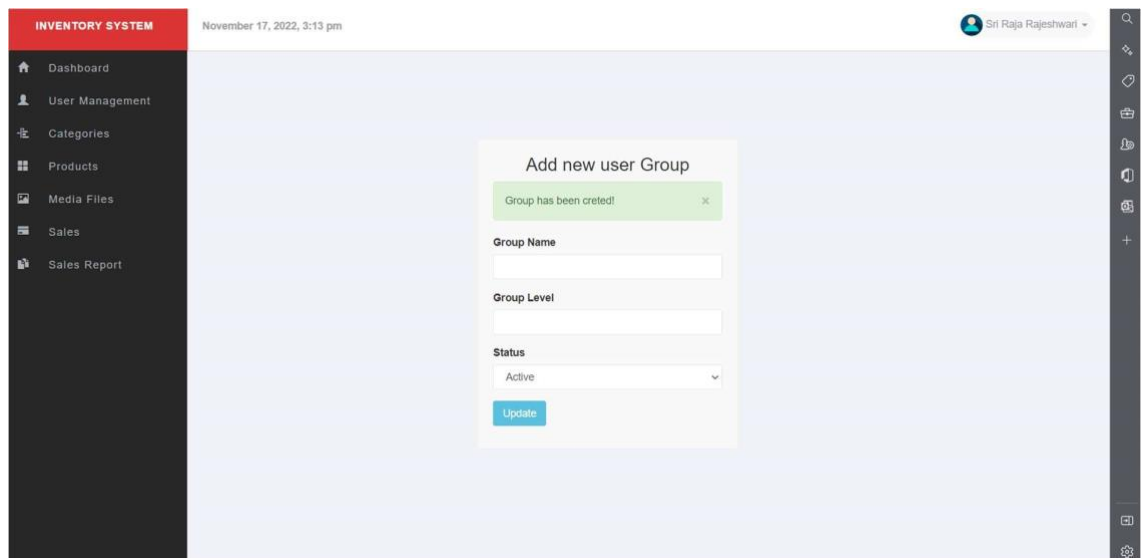


As an admin, I can add new group



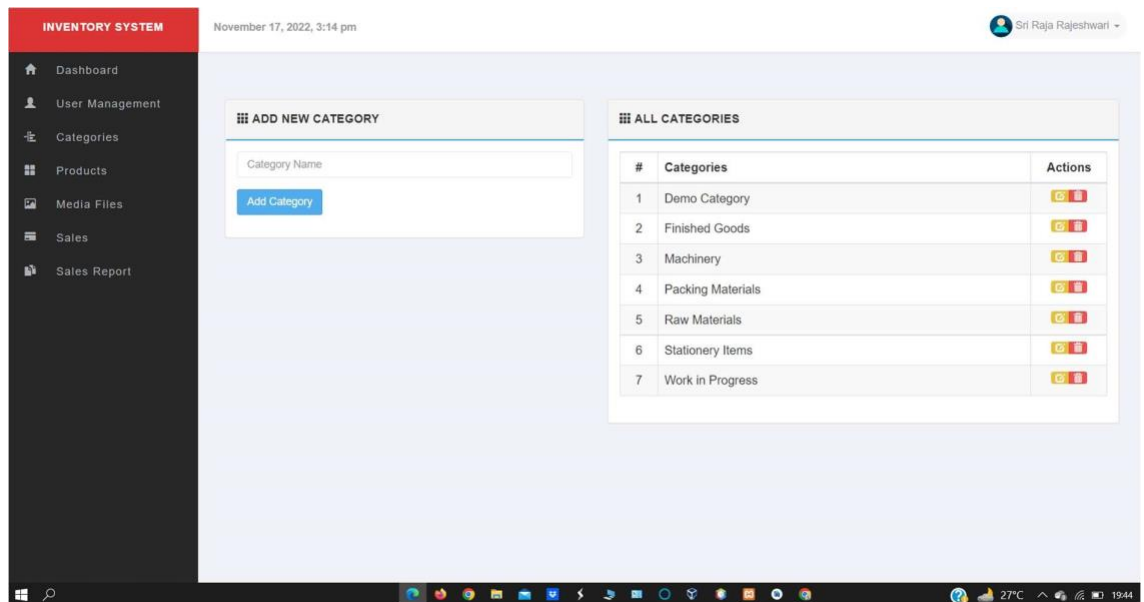


New user group has been created successfully

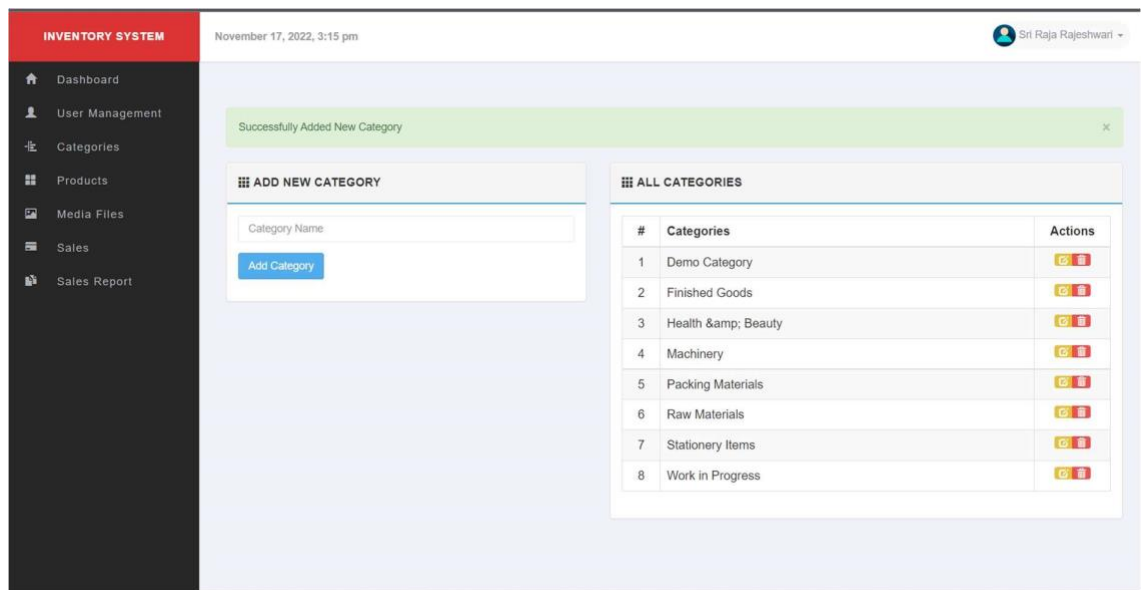


CATEGORIES

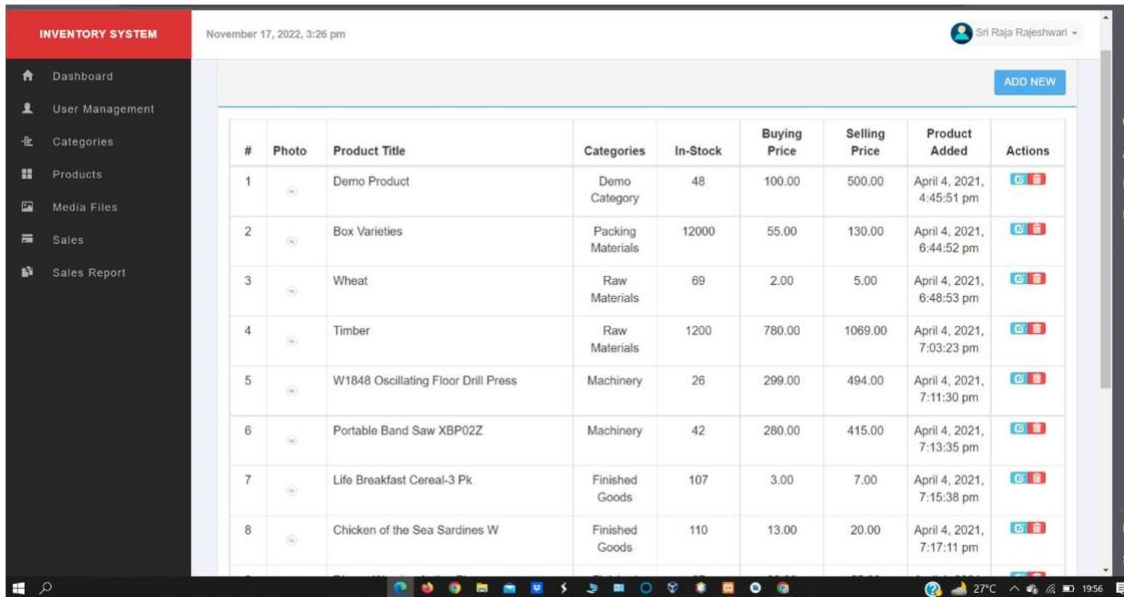
As an admin, I can add new category



New category has been created successfully



**As an admin, I can
add new product**

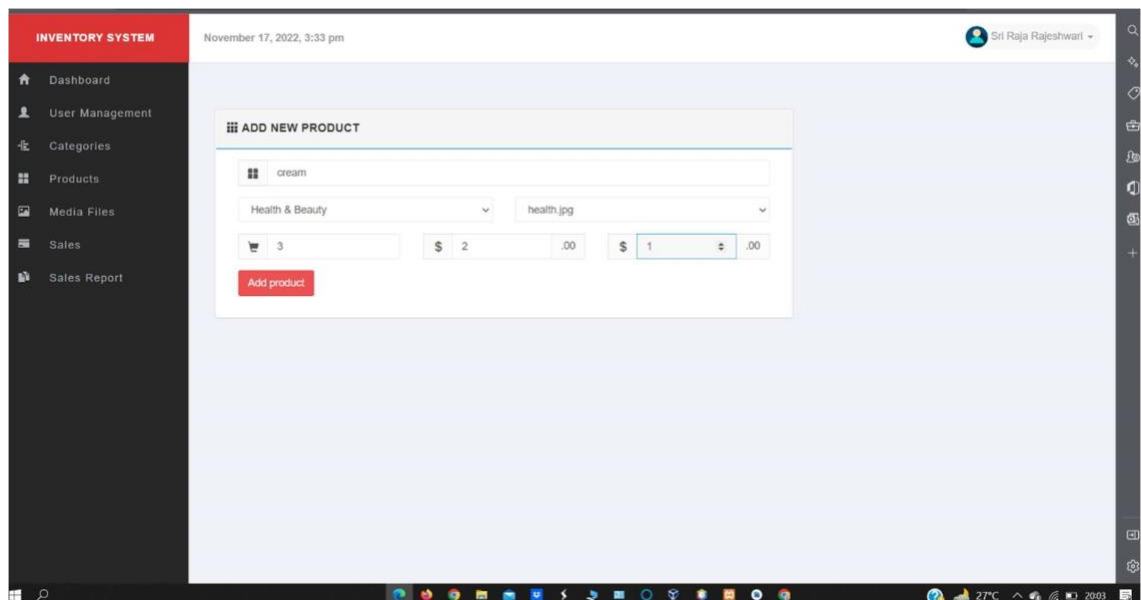


The screenshot shows the 'INVENTORY SYSTEM' dashboard. The left sidebar contains navigation links: Dashboard, User Management, Categories, Products, Media Files, Sales, and Sales Report. The main area displays a table of products with the following data:

#	Photo	Product Title	Categories	In-Stock	Buying Price	Selling Price	Product Added	Actions
1		Demo Product	Demo Category	48	100.00	500.00	April 4, 2021, 4:45:51 pm	
2		Box Varieties	Packing Materials	12000	55.00	130.00	April 4, 2021, 6:44:52 pm	
3		Wheat	Raw Materials	69	2.00	5.00	April 4, 2021, 6:48:53 pm	
4		Timber	Raw Materials	1200	780.00	1069.00	April 4, 2021, 7:03:23 pm	
5		W1848 Oscillating Floor Drill Press	Machinery	26	299.00	494.00	April 4, 2021, 7:11:30 pm	
6		Portable Band Saw XBP02Z	Machinery	42	280.00	415.00	April 4, 2021, 7:13:35 pm	
7		Life Breakfast Cereal-3 Pk	Finished Goods	107	3.00	7.00	April 4, 2021, 7:15:38 pm	
8		Chicken of the Sea Sardines W	Finished Goods	110	13.00	20.00	April 4, 2021, 7:17:11 pm	

An 'ADD NEW' button is located at the top right of the table.

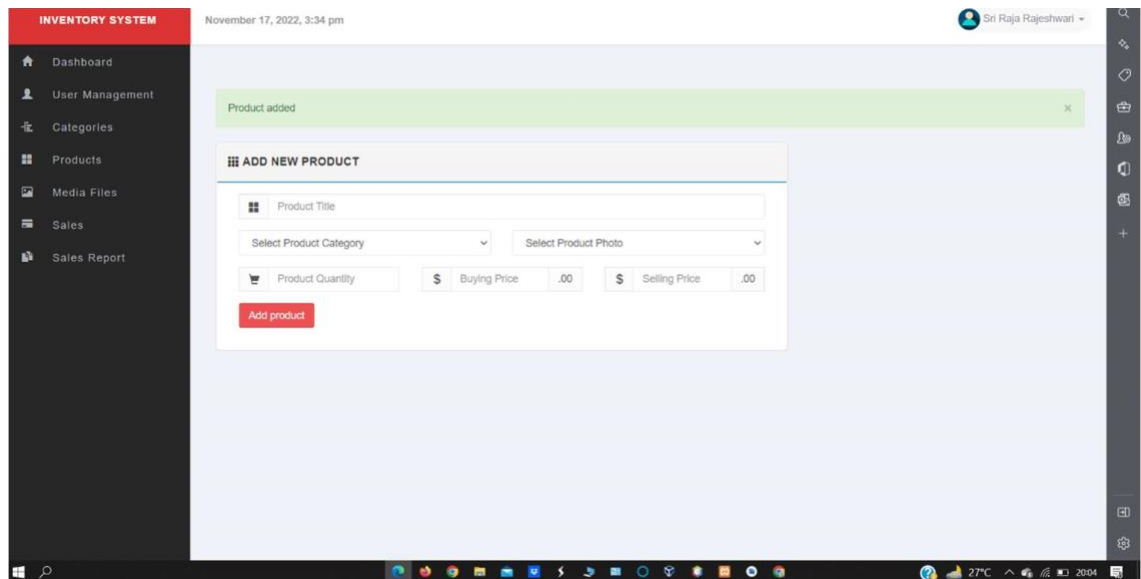
**As an admin, I can
select new product**



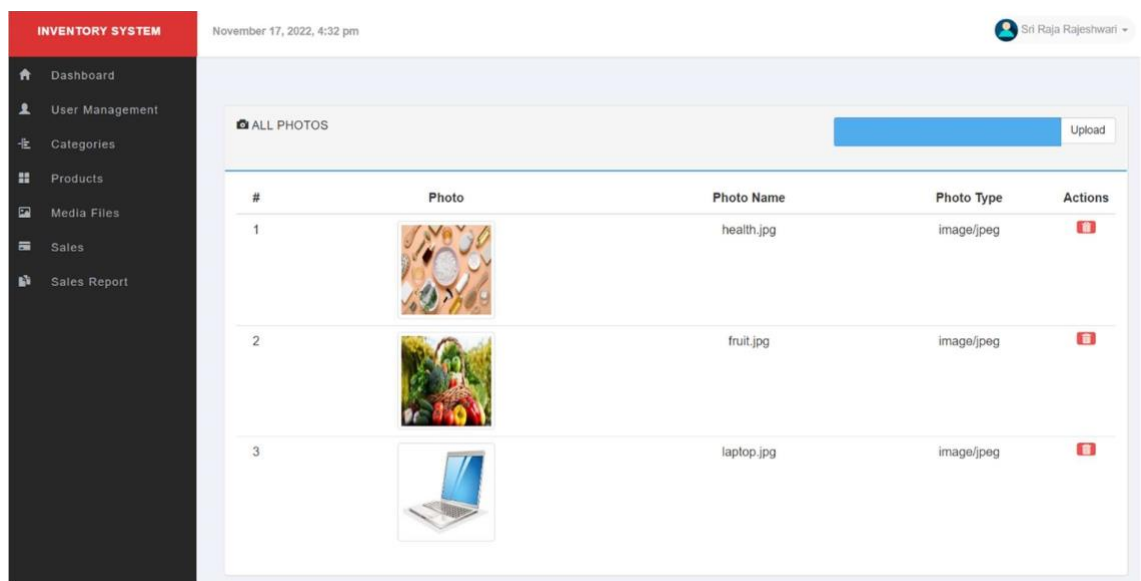
The screenshot shows the 'ADD NEW PRODUCT' form. The form includes the following fields and options:

- Product Name:** A text input field containing the word 'cream'.
- Category:** A dropdown menu with 'Health & Beauty' selected.
- Image:** A dropdown menu with 'health.jpg' selected.
- Quantity:** A text input field containing the number '3'.
- Buying Price:** A text input field containing '\$ 2 .00'.
- Selling Price:** A text input field containing '\$ 1 .00'.
- Action:** A red button labeled 'Add product'.

Product was added successfully



As an admin, I can manage the media files



















As an admin, I can manage the sales

INVENTORY SYSTEM November 17, 2022, 5:31 pm Sri Raja Rajeshwari

Dashboard User Management Categories Products Media Files Sales Sales Report

ALL SALES ADD SALE

#	Product name	Quantity	Total	Date	Actions
1	Demo Product	2	1000.00	2021-04-04	 
2	Wheat	3	15.00	2021-04-04	 
3	Hasbro Marvel Legends Series Toys	6	1932.00	2021-04-04	 
4	Portable Band Saw XBP02Z	2	830.00	2021-04-04	 
5	Classic Desktop Tape Dispenser 38	5	50.00	2021-04-04	 
6	Small Bubble Cushioning Wrap	21	399.00	2021-04-04	 
7	Life Breakfast Cereal-3 Pk	5	35.00	2021-04-04	 
8	Disney Woody - Action Figure	2	110.00	2021-04-04	 





As an admin, I can edit the sales

INVENTORY SYSTEM November 17, 2022, 5:36 pm Sri Raja Rajeshwari

Dashboard User Management Categories Products Media Files Sales Sales Report

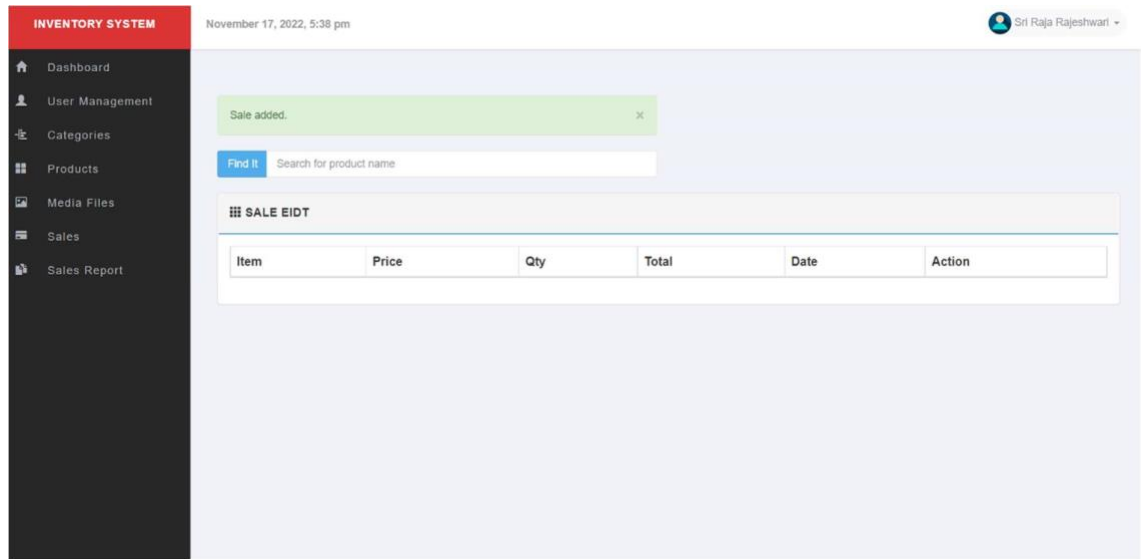
Find it: Chicken of the Sea Sardines W

SALE EDIT

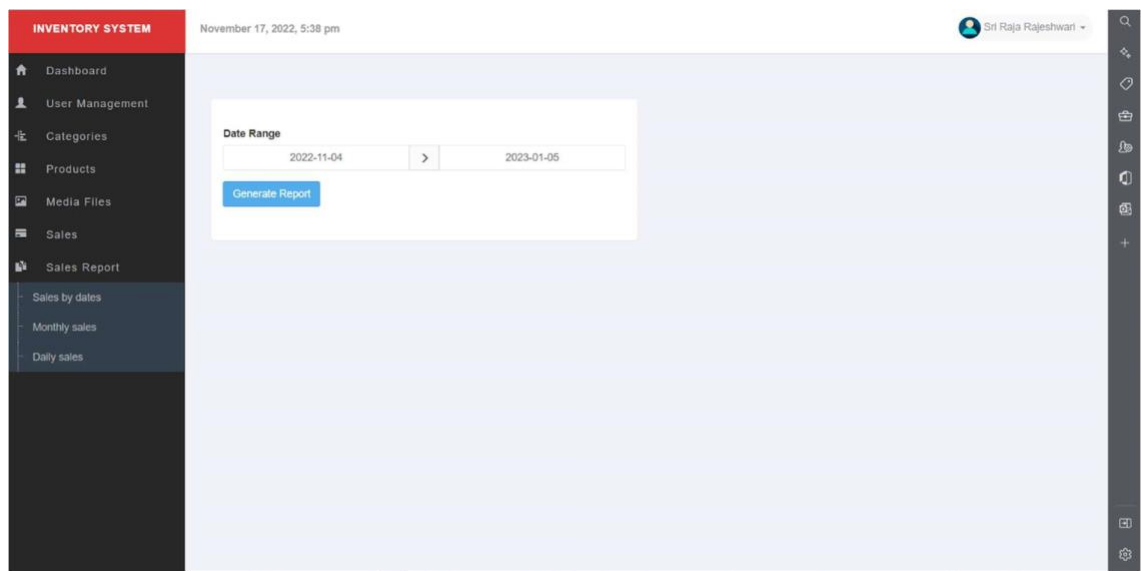
Item	Price	Qty	Total	Date	Action
Chicken of the Sea Sardines W	20.00	<input type="text" value="1"/>	20.00	17-11-2022 	  

Add sale

Sale was added successfully



As an admin, I can manage the sales report based on date range



As an admin, I can manage the sales report on monthly basis

INVENTORY SYSTEM November 17, 2022, 6:15 pm Sri Raja Rajeshwari

MONTHLY SALES

#	Product name	Quantity sold	Total	Date
1	Chicken of the Sea Sardines W	2	40.00	2022-11-17

As an admin, I can manage the sales report on daily basis

INVENTORY SYSTEM November 17, 2022, 6:16 pm Sri Raja Rajeshwari

DAILY SALES

#	Product name	Quantity sold	Total	Date
1	Chicken of the Sea Sardines W	2	40.00	2022-11-17

8. RESULTS

8.1 Performance Metrics

Inventory Performance is a measure of how effectively and efficiently inventory is used and replenished. The goal of inventory performance metrics is to compare actual on-hand dollars versus forecasted cost of goods sold. Many Lean practitioners claim that inventory performance is the single best indicator of the overall operational performance of a facility.

9. ADVANTAGES & DISADVANTAGES

- Paper-based retail inventory management can take a lot of time and effort. The retail inventory management software can cut short your in-store inventory process cycles through automation. Automation would give you time to focus on other productive business tasks.
- Inventory management is one of the crucial retail processes. Thus, any discrepancy in the inventory control would impact all other operations in your company. The retail inventory software can streamline the inventory processes, which would, in turn, improve the efficiency of your entire business
- Manual inventory control would increase your labor and process costs. The software would not only help you save time, but it would also help you reduce costs. As a result, the profitability of your business would improve. Also, you can invest the excess funds in activities that promote your business growth.
- One of the biggest problems with any computerized system is the potential for a system crash. A corrupt hard drive, power outages and other technical issues can result in the loss of needed data. At the least, businesses are interrupted when they are unable to access data they need. Business owners should back up data regularly to protect against data loss.
- Hackers look for any way to get company or consumer information. An inventory system connected to point-of-sale devices and accounting is a valuable resource to hack into in search of potential financial information or personal details of owners, vendors or clients. Updating firewalls and anti-virus software can mitigate this potential issue.
- When everything is automated, it is easy to forego time-consuming physical inventory audits. They may no longer seem necessary when the computers are doing their work. However, it is important to continue to do regular audits to identify loss such as spoilage or breakage. Audits also help business owners identify potential internal theft and manipulation of the computerized inventory system.

10. CONCLUSION

Inventory management is a very complex but essential part of the supply chain. An effective inventory management system helps to reduce stock-related costs such as warehousing, carrying, and ordering costs. As you have read above, there are different techniques that businesses can utilize to simplify and optimize stock management processes and control systems.

11. FUTURE SCOPE

In summary, successful companies will embrace the challenges of inventory management in the 21st century by leveraging the technology that is being offered through the Fourth Industrial Revolution. More important, companies will look at inventory as a strategic asset, that when properly deployed will deliver increased value and competitive advantage. Effective collaboration between supply chain partners will take on increased importance. The intensifying risks inherent with global sourcing in combination with a better appreciation of TCO will motivate companies to rethink their global inventory strategies.

12. REFERENCES

- Aggarwal, S.: A review of current inventory theory and its applications. *International Journal of Production Research* 12, 443–472 (1974)
- Anily, S., Federgruen, A.: One warehouse multiple retailer systems with vehicle routing costs. *Management Science* 36, 92–114 (1990)
- Beckmann, M.: An inventory model for arbitrary interval and quantity distributions of demand. *Management Science* 8, 35–57 (1961)
- Hamann, T., Proth, J.: Inventory control of repairable tools with incomplete information. *International Journal of Production Economics* 31, 543–550 (1993)